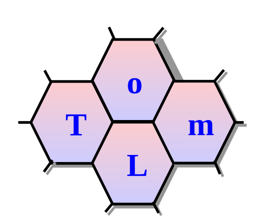
Der eigene Email-Server ….

…. eine Projektbeschreibung und Anleitung zum selber bauen



November 2023 (V. 4.3.2)

Inhaltsverzeichnis:

1. Einleitung

In der Vergangenheit hatte ich bei der Einrichtung von Softwarepaketen, die eine umfassende Einarbeitung ins Customizing erfordern, häufiger den unschönen Effekt erst nach mehrstündiger Arbeit festzustellen, dass diese Problemlösung am Ende entweder gar nicht so Recht zu meinem Problem passte oder dass sie meine Vorstellungen darüber, wie das sachliche Problem hier technisch mit Hilfe eines Programms gelöst werden sollte, nicht erfüllen konnte.

Deshalb denke ich heute, manchmal ist es schlichtweg nicht ausreichend, einfach nur zu beschreiben, wie man ggf. notwendige Pakete zu einem Themenkomplex mit „apt" installiert und welche Parameter in den zu den Paketen gehörenden Confs mit Werten zu setzen sind. Eine solche technisch-chronologisch-hierarchische Fakten-Auflistung konnte mir noch nie wirklich beim Verstehen der Zusammenhänge helfen. Insbesondere auch vor dem Hintergrund, dass es oft für ein sachliches Problem mehrere Programm-Alternativen mit unterschiedlichen Stärken, Schwächen und Fähigkeiten gibt, die eben mal mehr oder weniger auf die eigenen Rahmenbedingungen oder Vorstellungen passen. Darüber hinaus können diese verschiedenen Programm-Alternativen manchmal auch noch flexibel kombiniert werden.

Zum Schluss sollte man auch nicht vergessen, dass zu den gegebenen Rahmenbedingungen neben dem eigentlichen Problem selbstverständlich auch die eigenen Fähigkeiten gehören. Denn letztendlich sind es die eigenen Fähigkeiten die beschränken, was man selber umsetzen kann. Eine gute Anleitung kann allerdings die eigenen Möglichkeiten deutlich nach oben erweitern. Wie will man also ganz am Anfang entscheiden, mit welcher Programminstallation man am Besten seine eigenen Ziele erreicht und ob man das selber überhaupt umsetzen kann? Ein gute Anleitung kann bei der Beantwortung dieser Frage auf jeden Fall sehr hilfreich sein.

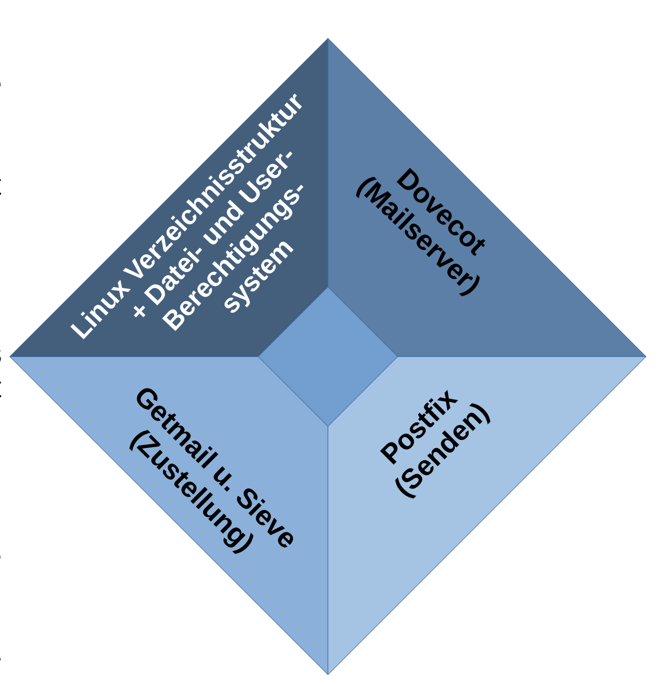
Gerade die am Anfang aus Unkenntnis getroffenen falschen Entscheidungen sind diejenigen, die zurück betrachtet meist viel sinnlose Arbeit ausgelöst haben…. und vielleicht manchmal auch viel Frust. Ich habe mir deshalb und nach den Erfahrungen bei der Installation meines Mailservers den Versuch vorgenommen, solche Effekte mit dieser Anleitung zu vermeiden. Deshalb ist diese Bauanleitung auch nicht nur eine Auflistung technischer Fakten und in bestimmte Reihenfolge gebrachte Einzelanweisungen, sondern zuallererst die Beschreibung eines Problems, zwar eines privaten Problems, welches aber dennoch durchaus auch als Beschreibung für viele „gleichgesinnte" Interessenten geeignet ist. Dann die Festlegung bezüglich der einzusetzenden Software und erst zum Schluss die tatsächliche Einrichtung und Inbetriebnahme. Diese Anleitung ist also aus der Sicht eines Anwenders entstanden und beschreibt die Lösung eines Komplexes von Problemen und Anforderungen – von der Idee bis zur Inbetriebnahme.

Ein ergänzender Hinweis in eigener Sache. Möglicherweise fehlen an einzelnen Stellen und zu besonderen Sachverhalten Erläuterungen darüber, warum so, was, wie und weswegen überhaupt. Das liegt dann daran, dass ich dann dafür kein Erklärung habe und diesen Sachverhalt aus irgendeiner Quelle übernommen habe und so als gegeben akzeptiert habe. Deshalb, weil es mit funktioniert, ohne jedoch nicht oder nicht wie gewünscht. Das ist aber einer der Gründe, die dabei helfen, diese Anleitung einzuschätzen und sie als das zu sehen, was sie letztendlich tatsächlich ist. Sie ist sicherlich nicht perfekt und sie ist keine Empfehlung, ein solches Projekt genau so zu realisieren, sondern nur eine Hilfe von einem Laien für Laien.

Meine Motivation für dieses Projekt war vielfältig. Zum Beispiel empfand ich seit einiger Zeit den bei uns langjährig etablierten Status Quo des Mailhandlings bei individueller Verwendung mehrerer Client-Geräte mittlerweile als mangelhaft – weil Thunderbird leider nicht mehrplatzfähig ist oder im Netz betrieben werden kann. Darüber hinaus suche ich immer nach Möglichkeiten, die Distanz zwischen mir/uns und den großen Internet-Konzernen zu vergrößern und mehr Autonomie zu erreichen - Google, Facebook, Whatsapp sind für uns ältere Anwender kompromisslos unerwünscht. Außerdem habe ich wirklich Spaß an der Entwicklung solcher Projekte, und noch mehr, wenn es gleichzeitig auch noch ein gewinnbringendes Ziel gibt…. und ja, es macht mir auch Spaß, eine solche Anleitung zu schreiben.

Abschließend möchte ich aber noch eins unzweifelhaft feststellen: Die Motivation zu dieser Lösung entstand aus einem rein privaten Interesse und meine Zielsetzung für diese Anleitung beschränkt sich allein auf den privaten Einsatz. Wenn diese Anleitung auch bei anderen, größeren Vorhaben hilft, ist das für mich wirklich OK, aber das bewerte ich dann als zufälligen Effekt.

Im Laufe dieser Dokumentation erwarten den künftigen Mail-Server-Administrator 4 komplexe Bereiche, die abgearbeitet bzw. bearbeitet werden müssen und die man allesamt und jeden für sich als ziemlich umfangreich und teilweise auch kompliziert einschätzen kann. Ich werde versuchen, durch Kontinuität bei den Bezeichnungen und Namen immer eine gerade Linie zu verfolgen und zu bewahren. Gerade was Bezeichnungen und Namen angeht, machen es Holprigkeit und ständiges Wechseln und immer wieder neue Namen fast unmöglich, die Zusammenhänge als Ganzes zu verstehen.



Ganz sicher werden erfahrene Administratoren wegen des Umfangs dieser Anleitung genervt sein, aber für diese habe ich das nicht geschrieben. Die Zielgruppe sind Anwender, wie ich selber, die wir alle keine Administratoren sind. Aber gerade wir Nicht-Administratoren sollten uns bei einem solchen EDV-Projekt schon wirklich darüber im Klaren sein, was wir hier überhaupt tun. Und das, was wir hier tun, sollte wenigstens in einem Mindestmaß auf Verstehen beruhen. Irgendeine halbwegs (un)passende Web-Anleitung per Copy/Paste unreflektiert auf den eigenen Server zu schreiben und die Dienste einfach zu starten, ohne zu wissen, was da passiert, warum und wie, ist meiner Einschätzung nach die schlechteste Vorgehensweise. Das ist der auch Grund für meinen vielen Erklärungen, denn neben den technischen Zusammenhängen gibt es immer auch sachliche Zusammenhänge … und das wichtigste ist, man muss das alles auch wirklich verstehen.

Der richtige Weg ist meiner Meinung nach, bevor man anfängt irgendwelche Software zu installieren, zuerst das Problem in allen Facetten zu erfassen, danach eine mögliche Lösung zu skizzieren, als nächstes die passenden Werkzeuge zusammenzustellen und als letztes alles zielgerichtet und planmäßig zusammenzubauen. Also bin ich es irgendwann angegangen, zuerst mit dem Versuch, einmal die familiären IT-technischen Probleme beim Email-Handling anschaulich darzustellen…...

Wo ist also das Problem? ***Wir*** sind das Problem! ***Wir*** sind die Addams-Family…

- Thomas Vlad.D. Addams
- Silvia N. Addams
- Manuel N. Addams
- Steffi J. Addams
- Bibi N. Addams
- Marco R. Addams … und noch viele viele weitere Familienmitglieder in unserer Burg.

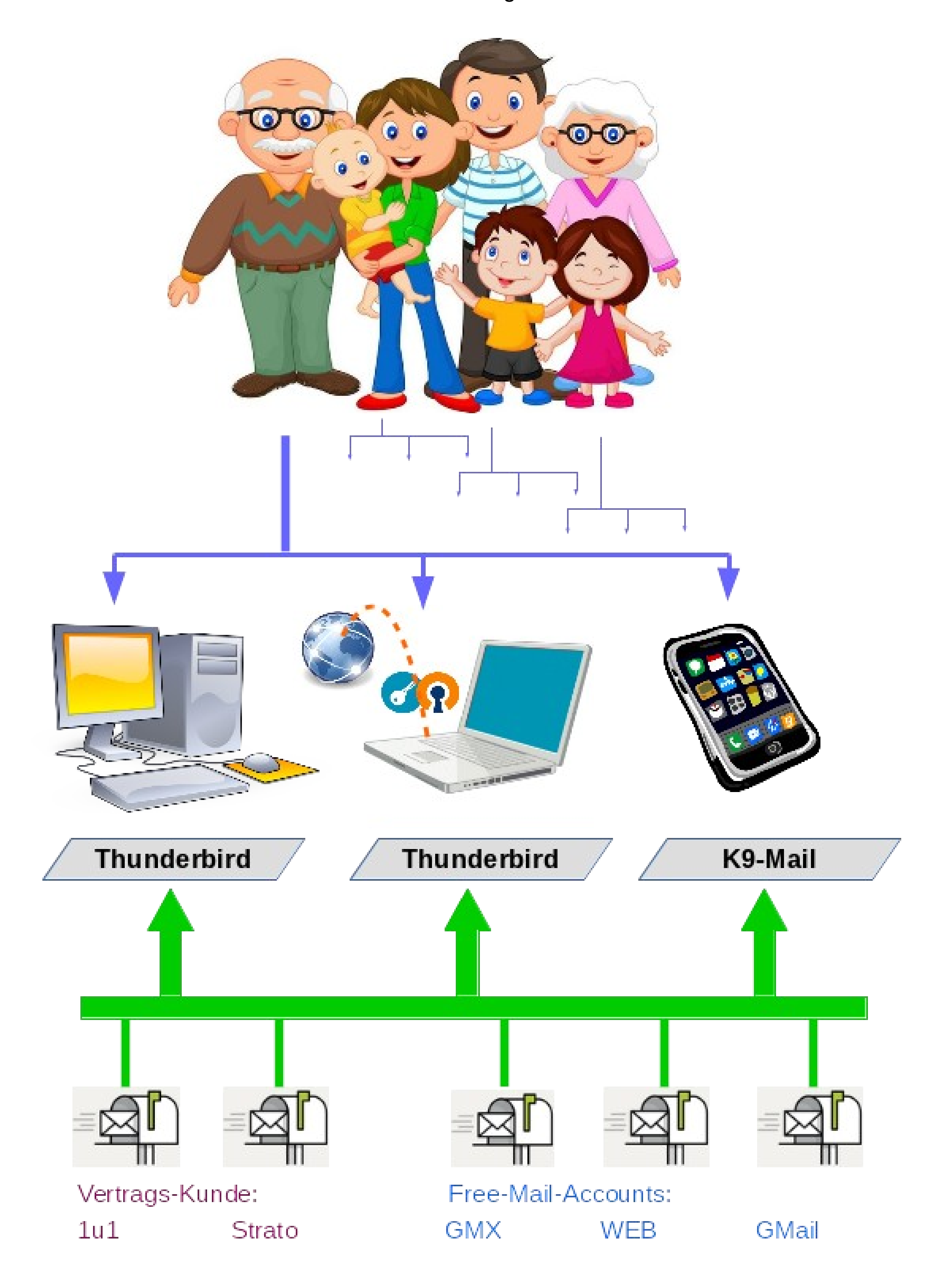
Die folgende Tabelle enthält eine Bestandsaufnahme und Überblick über die Art unserer derzeitig genutzten Email-Adressen, vertraglich und z.T. mit Real-Namen personifiziert und weiterhin Freemail-Accounts mit Fake-Names. Hinweis: Eine zufällige Übereinstimmung mit realen Adressen ist nicht beabsichtigt, sondern wirklich ein Zufall. Solche Beispieladressen sind für das Verstehen der Zusammenhänge allerdings unerlässlich.

Thomas	thomas.addams@toml.de thomas.addams@gmx.de toml@web.de zorro-ohne-muetze@example.com
Silvia	silvia.addams@toml.de silvia.addams@gmx.de wonderwoman@web.de hotlady1924@example.com
Manuel N. Addams	manuel.addams@toml.de manuel.addams@gmx.de low-performer@web.de
usw. für die anderen Mitglieder	

2. Das Problem - eine Zusammenfassung

- Eine Familie, in der Computer für alle Mitglieder ein Bestandteil des alltäglichen Lebens sind
- Jedes Familienmitglied verfügt über mehrere typische Client-Geräte
- Die Familienmitglieder haben alle mehrere Email-Accounts
- Die Email-Konten sollen idealerweise auf allen Client-Geräten verfügbar sein

Eigentlich könnte man feststellen, wenn man auf das Bild schaut, dass das völlig normale Umstände, Gegebenheiten und Rahmenbedingungen sind – so wie sie wahrscheinlich sehr ähnlich in vielen Familien existieren. Warum also nicht einmal eine Lösung dafür suchen?



Die weiteren Fakten sind: Die Familie hat einen Vertrag mit einem Internet-Provider für den Internetzugang und darüber hinaus eine eigene Domain bei einem Webspace-Hoster, auf dem auch Postfächer mit der eigenen Domain eingerichtet sind. Aus der Vielzahl der ISP-Angebote habe ich jetzt hier einfach mal stellvertretend für alle anderen Provider 1&1 und Strato genannt. Dem sollte aber keine besondere Bedeutung beigemessen werden, es sind einfach die, die mir eingefallen sind. Es existieren also gleichzeitig für die Familienmitglieder sowohl vertragliche Email-Adressen, die durchaus auch namentlich personifiziert sein können, als auch (ich nenne sie mal Fake-Adressen) Email-Adressen bei Free-Mail-Hostern.

An diesem Punkt angekommen könnte man einfach feststellen, dass alle Mail-ISP mit dem Angebot der IMAP-Accounts doch heute schon genau diese Punkte absolut zufriedenstellend ermöglichen. Ja, richtig, technisch betrachtet ist das möglich und wohl auch zufriedenstellend – aber leider nicht aus einer sachlichen Perspektive… es gibt nämlich noch andere Rahmenbedingungen, als nur die technischen.

In diesem Konzept betrachte ich hier Probleme oder einzelne Aspekte der Einfachheit halber immer aus der Perspektive einer Person, und zwar dem Opa, der ganz zufällig auch Thomas heißt, dessen Sicht aber trotzdem stellvertretend auch für alle anderen gilt.

3. Rahmenbedingungen: eigene Vorstellungen, weitere Anforderungen

Zu den den oben genannten Anforderungen gibt es noch weitere spezielle Anforderungen, ich nenne sie hier mal Rahmenbedingungen, die nicht so einfach oder nur mit Mängeln mit mehrfachen IMAP-Accounts gelöst werden können.

- Opa ist ein Reisefan und möchte mit seinem Laptop auch auf seinen Reisen einen sicheren Zugang zu seinen Postfächern haben, auch an potentiell unsicheren Accesspoints. Seine Verbindung nachhause wird via OpenVPN aufgebaut.

- Wichtige Emails aus seinen verschiedenen Postfächern möchte Opa an einer für ihn zentralen Stelle in einem zusammenfassenden Mail-Archiv speichern können.

- Emails mit dem Charakter „*Geschäftlich*“, „*Dienstlich*“ oder „*Betrifft die ganze Familie*“ möchte Opa in ein von allen gemeinsam nutzbaren Archiv speichern.

- Die 3 vorherigen Punkte treffen auch auf die anderen Familienmitglieder zu.

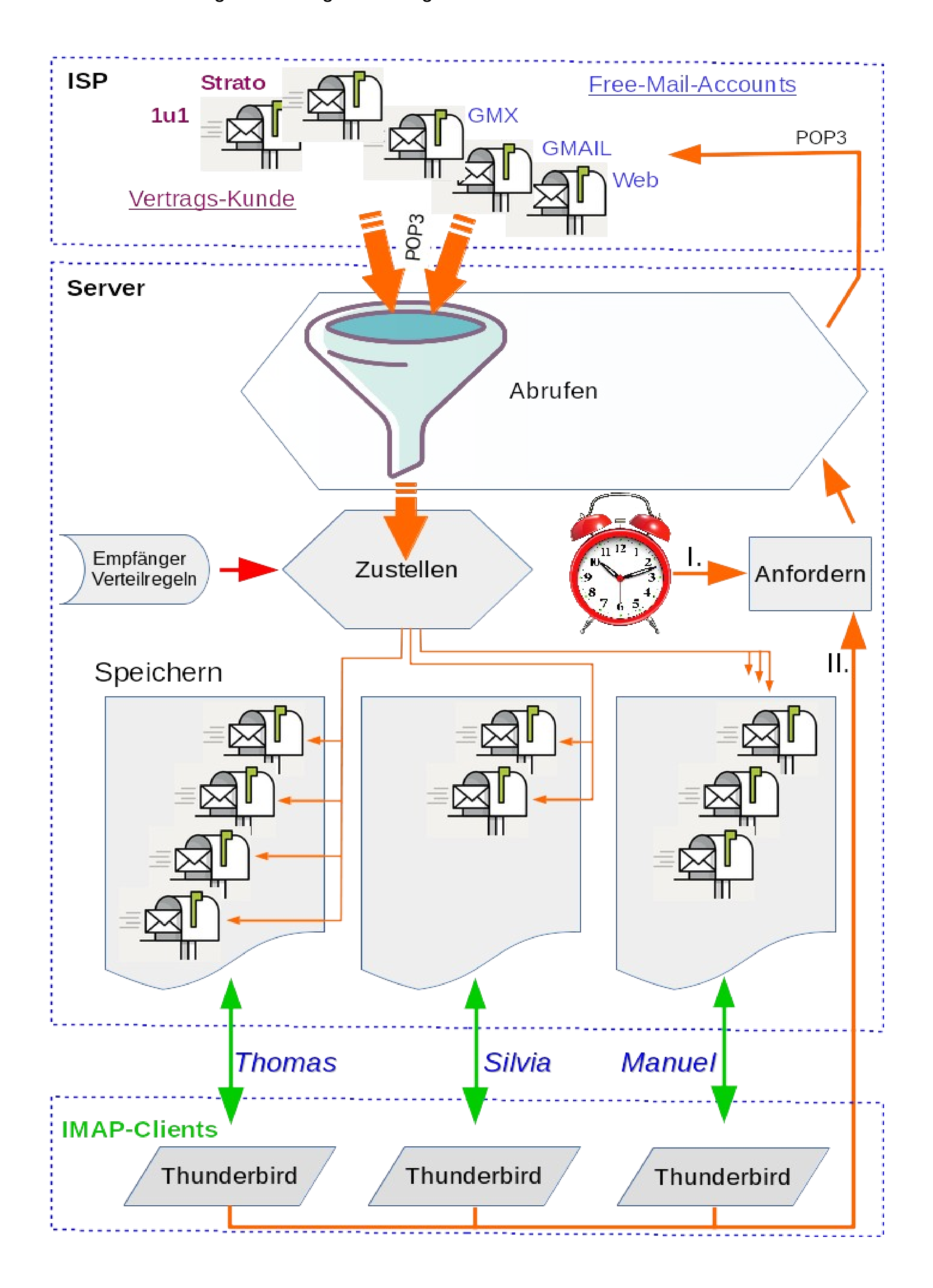
- Alle Familienmitglieder stimmen darin überein, dass sie den Familien-Geschäftlichen Schriftverkehr, möglicherweise mit vertraulichen, sensiblen, schutzwürdigen persönlichen Daten, oder Schriftverkehr mit Banken, Finanzamt, Behörde, oder einfach nur privater vertraulicher Schriftverkehr mit Verwandten und Freunden, usw. auf gar keinen Fall dauerhaft und sich ständig vermehrend auf dem Server eines fremden Unternehmens hinterlegen wollen, dessen Geschäftsziel die Auswertung und der Handel mit den persönlichen Daten der Familienmitglieder ist.

  Das betrifft insbesondere die Archive, ganz egal ob persönlich oder gemeinsam, weil eben genau in diesen Archiven die schutzwürdigen Daten aus mehreren Postfächern verschiedener Familienmitglieder eventuell sogar langfristig gespeichert werden und dadurch immer umfangreicher werden. Die Prämisse lautet also: Eigene „echte“ Daten sollen nicht dauerhaft fremd gespeichert werden.

- Der letzte Punkt, für mich einer der wichtigsten, betrifft die Datensicherheit. Nicht jeder, der einen Home-Server betreibt, ist gleichzeitig auch wirklich ein geschulter und sachkundiger Administrator. Der gehackte eigene Server, digitaler Einbruch und Diebstahl unserer Daten, der Missbrauch unseres Servers z.B. als Spambot, die Ausspähung unseres datentechnischen Lebens … all das muss effektiv verhindert werden. Aus diesem Grund soll der Email-Server selber nicht von außen aus dem Internet zugänglich sein. Auf die damit einhergehenden Restriktionen werde ich im weiteren Verlauf noch gesondert eingehen.

4. Drei mögliche Lösungsmodelle

Bei einer ersten Überlegung bieten sich sofort zwei Modelle an, in dem entweder die Emails von allen Email-Accounts aller Familienmitglieder regelmäßig maschinell abgeholt und auf dem heimischen Server gespeichert werden. Oder alternativ, dass jedes Familienmitglied mit dem Start des Email-Clients Thunderbird die vollständige Abholung seiner eigenen Emails selber veranlasst.



Betrachten wir kurz die Vor- und Nachteile dieser zwei Modelle.

Modell 1:

Der Email-Server holt in einem festgelegtem Zyklus, z.B. alle 15 Minuten, die Post aus sämtlichen Postfächern ab und verteilt alle Emails an die lokalen Postfächer der einzelnen Familienmitglieder. Das funktioniert gut, hat aber auch ein paar gravierende Nachteile.

- ✖ Weil unser Mailserver nicht von außerhalb über das Internet erreichbar sein soll, würden wir unterwegs und über die Funktion „Mobile Daten“ unseres Smartphones die Emails nicht mehr beim direkten Zugriff auf die ISP-Postfächer sehen können. Deshalb nicht, weil der Server regelmäßig alle Emails abholt und sie dabei auf dem fremden Server -wie von uns erwünscht- sofort löscht.

- ✖ Gleichzeitig werden dabei auch die Free-Mail-Accounts (mit den Fake-Adressen) abgeholt, was möglicherweise regelmäßig haufenweise unerwünschte Spam-Mails in unsere Server-Datenbank reinspült.

Modell 2:

Die Abholung der Emails wird dynamisch beim Start des Clients (bei uns ist das Thunderbird) am heimischen PC veranlasst. Das heißt, solange die Mails nicht abgeholt wurden, können sie vollständig von allen Mobil-Geräten mit einem Mailprogramm auf dem Smartphone (z.B. K9-Mail) und einer direkten Anmeldung beim MAIL-ISP via IMAP geöffnet werden.

- ✖ Der obige Nachteil No. 2 bleibt aber auch hier erhalten. Auch hierbei werden bei der endgültigen Abholung der Free-Mail-Accounts ggf. haufenweise Spam in die Server-DB gespült. Das ist natürlich nicht wirklich tragisch, aber es nervt und macht Arbeit, weil jeder Anwender diesen Mist wieder manuell entfernen muss. Alternativ könnte ich unerwünschte Emails auch maschinell durch zusätzliche Tools/Plugins wie Greylisting, ClamAV, Spamassassin oder Sieve-Antispam entfernen lassen. Aber meine Devise ist immer schon und ohne jeden Zweifel *„weniger ist mehr!“*, und der möchte ich auch gerne hier folgen. Wenn es also Möglichkeiten gibt, die unerwünschten Emails mit einer durchdachten Herangehensweise erst gar nicht auf dem Server vorzufinden, muss ich anschließend auch nicht nach maschinellen Lösungen suchen, diese wieder automatisiert zu entfernen.

Letztendlich kam ich zu dem Fazit, dass diese zwei Lösungsansätze tatsächlich auch nur suboptimal sind und bin dann auf eine dritte Lösung gekommen, die ich nun erläutere und die in der folgenden Grafik skizziert ist.

Modell 3:

Der Lösungsansatz sieht nun wie folgt aus:

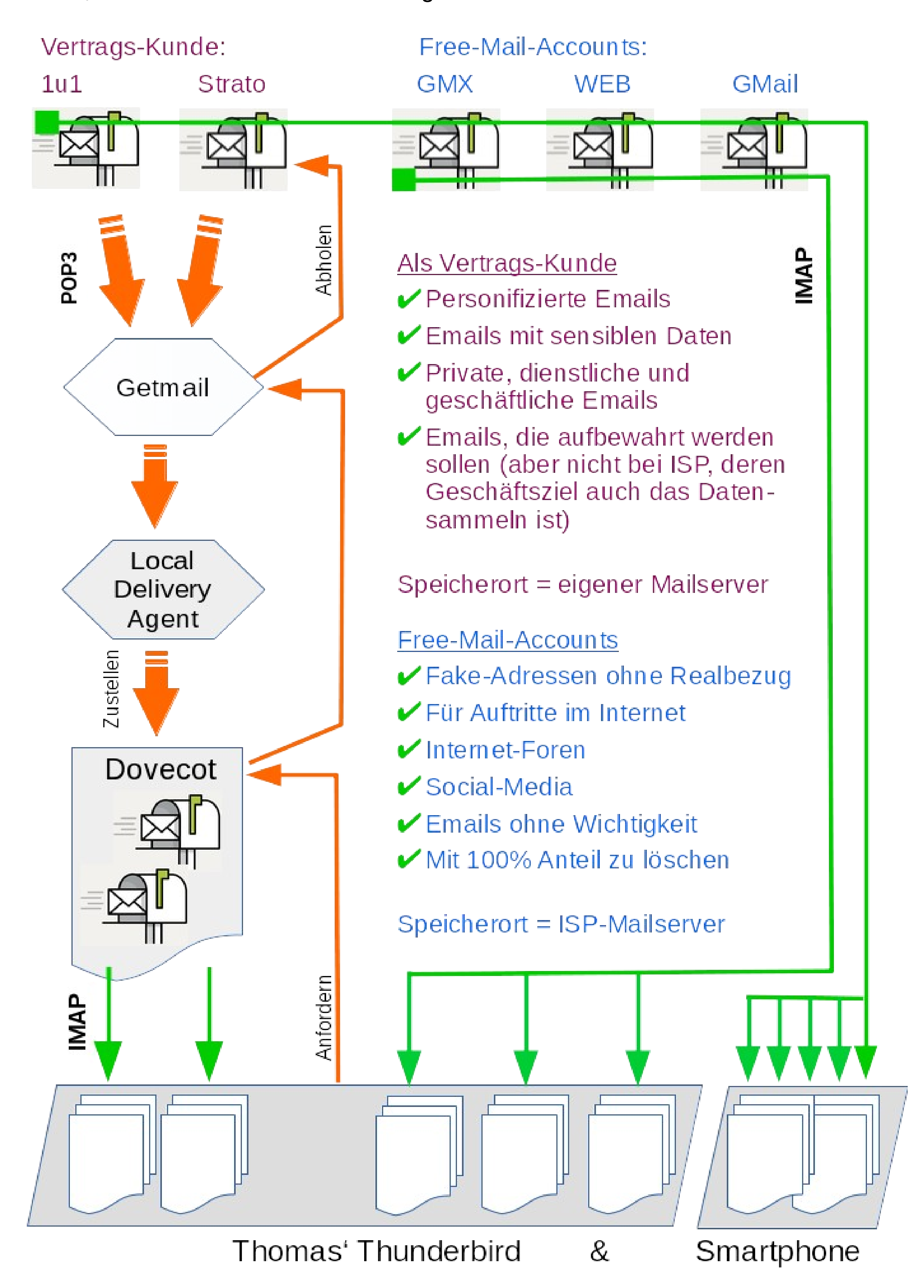
- ✔ Die Anwender initiieren die Abholung ihrer Emails selber, damit sie tagsüber und abwesend von Zuhause über das Smartphone mit direktem Zugriff auf die Postfächer des Mail-ISP trotzdem neu angekommene Emails via IMAP öffnen können.

- ✔ Es werden nur die personifizierten Email-Konten mit ausschließlich seriöser Post endgültig via POP3 abgeholt und in unserer Server-DB gespeichert.

- ✔ Vor dem Hintergrund, dass die Fake-Adressen-Postfächer sowieso keine wichtige und zur Aufbewahrung relevante Post enthalten und dort selbstverständlich auch keine sensiblen Daten existieren, werden diese Postfächer immer nur via Mail-ISP und IMAP bearbeitet. Darüber hinaus kann man in diesen Postfächern auch einfach regelmäßig alle unbedeutenden Mails löschen.

  Das bedeutet, ein Download in unsere Server-DB findet für diese unwichtigen und ggf. unerwünschten Emails gar nicht erst statt. Und trotzdem sind diese Postfächer immer sowohl von den Heim-PCs als auch über die Mobil-Geräte erreichbar. Spam-Emails, die wir gar nicht erst öffnen, oder die vom Mail-ISP bereits als Spam markiert sind, erreichen unseren Server also gar nicht.

Das Modell 3, welches letztendlich auch umgesetzt wurde:



Über die Thunderbird-Clients wird individuell bei Programmstart die Abholung der Emails veranlasst. Die Emails aller Konten sind deshalb solange per IMAP-Zugriff über das Smartphone zu öffnen, wie sie nicht zuhause am PC endgültig abgeholt wurden. Die Fake-Adressen werde nie abgeholt, sondern immer nur via IMAP direkt beim Mail-ISP geöffnet.

Welche der eigenen Email-Adressen als „seriös“ oder als „Fake“ eingestuft werden, muss man für sich selber entscheiden. Selbstverständlich kann auch eine GMX- oder GMAIL-Adresse personifiziert sein und weitestgehend für seriösen Schriftverkehr genutzt werden. Diese Entscheidung mit anschließender Festlegung ist flexibel und muss individuell getroffen werden.

5. Software-Alternativen

Nachdem nun festgelegt ist, welches Ziel überhaupt erreicht werden soll, muss nun entschieden werden, welche Software dazu am besten geeignet ist. Das ist wieder schwierig, weil das ganze Thema für mich komplettes Neuland war und ich anfangs keines der Pakete kannte und weil es für jeden einzelnen Prozess-Schritt auch noch verschiedene Alternativen gibt.

Email

- abholen — Fetchmail und Getmail
- verteilen u. zustellen — procmail, maildrop, Dovecot-LDA & sieve-Plugin
- versenden — exim, postfix
- lokal speichern — dovecot, dbmail

Es gibt weitere, aber diese genannten sind definitiv die Namen, die man über die Websuche immer wieder als erste findet. Und gerade für das Trio Dovecot + Exim + Fetchmail findet man eigentlich nur relativ gutes Feedback. Dovecot und Exim sind wohl beide jeweils der Klassenprimus in seiner Kategorie. Zu Exim wird im allgemeinen festgestellt, dass es unerreicht flexibel ist und wirklich alle Ansprüche erfüllen kann.

Bei diesen Voraussetzungen war natürlich Dovecot von Anfang an in meinen Überlegungen gesetzt. Und ich habe mich entschieden, wie so viele andere vor mir auch, mit Exim und Fetchmail anzufangen. Die Basisfunktionalität hat mit diesen Komponenten dann auch sehr schnell funktioniert. Für einen Testuser Emails abholen oder senden klappte auf Anhieb. Aber dann, mit den Einstellungen für das Senden von Emails durch mehrere User einschließlich notwendiger Plausibilitäts- und Berechtigungsprüfungen und der Festlegung anwenderbezogener logischer Richtlinien, sowie der grundsätzlichen Verschlüsselung des Traffics und der Benutzung nur mit virtuellen Anmeldedaten war ich hoffnungslos überfordert. Exim ist vermutlich durch kontinuierlichen funktionalen Wachstum im Laufe der Zeit in einem Maße umfangreich und hinsichtlich eines eher traditionellen Linux-Customizings-Styles so unübersichtlich geworden, dass es meine Fähigkeiten, das autodidaktisch sinnvoll und mit vertretbaren Aufwand anzuwenden, einfach überstiegen hat. Ich bin dann zu der Entscheidung gekommen, dass ich mit Exim nicht die Systemintegrität und Sicherheit erreichen kann, die ich mir vorstelle. Mit Exim würde ich -bei mit einem für private Anwendungen vertretbaren Arbeitsaufwand- meine Grenzen und Möglichkeiten überschreiten, dass gesamte Projekt auch später noch zu verstehen.

Ganz ähnlich war es gleichzeitig auch bei Fetchmail, was ich auch nicht auf den ersten Blick als nachvollziehbar verstanden habe. Als ich schließlich auch noch einige Kritiken sowie Hinweise auf anscheinend vorhandene Sicherheitslücken gelesen habe, habe ich diesen Versuch dann endgültig aufgegeben.

Mittlerweile merkbar irritiert über die Komplexität und anscheinende Kompliziertheit dieses gesamten Themas habe ich dann einen neuen Versuch mit Postfix als Ersatz für Exim und Getmail als Ersatz für Fetchmail gestartet. Schon unmittelbar beim Lesen im Web hat es mir gefallen, dass Sicherheit und einfaches Customizing die Prämissen bereits bei der Entwicklung von Postfix waren. Das gefiel mir wirklich, weil ich eigentlich immer versuche, sorgfältig mit dem Thema Sicherheit umzugehen. Und hinsichtlich des Customizings muss es natürlich für mich insofern beherrschbar sein, weil ich als Privat-Nutzer eben nicht ständig am System schraube und darüber immer tiefer in die Materie eindringe und mehr und mehr Kenntnisse erwerbe. Nun ja, mit Postfix funktionierte die Basisfunktionalität ebenfalls auf Anhieb. Aber Postfix hat es mir mit einem tollen Wiki dann auch bei den weiteren Schritten wirklich leicht gemacht, auch die zusätzlichen (o.g.) Anforderungen umzusetzen, an denen ich zuvor gescheitert bin.

Mein persönliches Fazit ist insofern natürlich eindeutig. Exim verfügt ganz sicher über einen Umfang, der auch eine Einrichtung nach professionellen Ansprüchen ermöglicht. Aber es ist für meine Bedürfnisse schlichtweg zu mächtig und deshalb unpraktisch, wenn ich von der Menge an vorhandener Funktionalität allenfalls nur einen Teil beanspruche, aber mir die pure andere Menge es ansonsten unmöglich macht, mein Mail-System mit Durchblick zu beherrschen. Die in Exim vorhandene Syntax für Einstellungen, Regeln und der Verwendung von (internen) Umgebungsvars sowie der internen Zusammenhänge, etc. war für mich einfach nicht intuitiv reproduzierbar.

Und genau diese beiden Neuausrichtungen, hin zu Postfix und Getmail, waren der Startschuss zu einer erfolgreichen Umsetzung ohne weitere große Probleme. Wer nicht eine immens umfangreiche und keine Wünsche offenlassende Flexibilität benötigt, die privat möglicherweise eh nur zu einem Bruchteil ausgenutzt wird und gar nicht den Anspruch hat, eine Installation zu realisieren, die einer professionellen Großlösung gleichkommt, ist mit Postfix und Getmail wirklich bestens bedient. Ich bin ganz sicher, dass auch Postfix die Fähigkeiten zu einer großen Lösung hat. Aber Postfix hat darüber hinaus noch etwas sehr wichtiges …. und zwar nach meinem Empfinden (das eines Computer-Laien) eine maßgeblich leichtere Bedienbarkeit mit eben intuitiven Formulierungen für meine doch eher einfachen Anforderungen in einer sehr überschaubaren Größenordnung. Im Postfix-Konzept ist enthalten, dass genau darunter eben nicht die Sicherheit leidet. Eine Sicherheit, die ich nicht erst umständlich definieren oder konzipieren muss ist eine der Prämissen von Postfix. Das jedenfalls hat mich absolut überzeugt.

Die letzte zu treffende Entscheidung fiel dann zunächst auf Procmail, welches die von Getmail abgeholten Emails nach meinen Regeln und Vorgaben auf die Benutzerpostfächer verteilen sollte. Procmail konnte ich in meiner ersten Mailserver-Installation relativ schnell erfolgreich zum Einsatz bringen. Procmail lief sogar richtig klasse. Bis ich dann durch einen Internet-Kontakt einen Hinweis bekam, dass der Entwickler oder Maintainer von Procmail empfiehlt, Procmail auf Grund seines Alters nicht mehr zu verwenden. Tja, gerade alles fertig und nun heißt es „G*ehe zurück auf Star*t“. Nach einigen nun anschließenden Überlegungen, ob ich vielleicht Maildrop als Nachfolger verwenden soll oder vielleicht die Zustellung via Postfix einrichte, habe ich mich am Ende entschieden, ein in Dovecot sowieso vorhandenes Feature zu nutzen – und zwar Dovecot LDA. Nach der Einarbeitung in dieses neue Feature, der Implementierung von Sieve, vielen Tests und einiges an Problembeseitigung kann ich jetzt mit Blick auf den aktuellen Stand meines Mailservers resümieren, dass das ein richtig guter Schritt war, der meine Installation mit einer die Aktion begleitende Neustrukturierung der Datenverzeichnisse nachhaltig verbessert hat.

Die derzeitige Software-Installation sieht so aus:

Mail-Server	➔	Dovecot IMAP
Mail-Transfer-Agent	➔	Postfix und Dovecot LMTP
Mail Retrieval Agent	➔	Getmail
Mail-Delivery-Agent	➔	Dovecot LDA und Sieve

6. Voraussetzungen – die berechtigten Anwender

Bevor es nun an die Installation und Einrichtung der Programme geht, muss ich noch auf eine Voraussetzung hinweisen. Alle User, die später den Mailserver nutzen sollen, müssen auf dem Server als Linux-User eingerichtet sein. Der Mailserver selber verwendet zwar nur Anmeldedaten virtueller User, aber meine Vorgabe, die Abholung der Mails nicht zyklisch via Cron durchzuführen, sondern anwenderindividuell und ad hoc bei Bedarf, erfordert, dass bestimmte Prozesse unter der UID des Anwenders ablaufen. In diesem Fall ist es das Duo Getmail/LDA, welches ein setuid auf den Anwender durchführt, für den es die Emails in den Postfächer schreibt. Das hat den Vorteil, dass wirklich alle Dateien im Mail-Ordner des Anwenders dem Anwender selber gehören. Somit hat neben dem Anwender selber wirklich nur noch „root“ ein Recht, diese Ordner zu öffnen.

Nach meiner Überzeugung sollte auf einem privaten dedizierten Server, der im Regelfall auch nur von einer einzigen Person administriert wird, kein User existieren, der berechtigt ist, direkt „*als er selber*“ Änderungen am System durchzuführen. Nicht einmal die Person, die das root-Password kennt. Der bei einigen Distributionen praktizierte Weg, den ich für geradezu fahrlässig leichtsinnig und zweifelhaft erachte, normalen Usern solche Rechte einzuräumen, ist der Befehl „sudo“. Ich persönlich halte „sudo“ für völlig unnötig und überflüssig und durch den Password-Keep auch für ein unkalkulierbares Sicherheitsrisiko. Aus diesem Grunde werde ich hier mit keinem Wort auf die Verwendung „sudo“ oder einer der grafischen (genauso überflüssigen) Prothesen eingehen. Ich gehe davon, dass es einen root-Account mit Password gibt und das der Administrator sich als root anmelden kann, um die komplette Einrichtung nicht als er selber, sondern als root durchzuführen.

7. Installation und Vorbereitung der Inbetriebnahme

Die Installation der benötigten Pakete ist einfach und schnell von Hand im Terminal gemacht. Es ist notwendig, mit der Installation der benötigten Pakete zu beginnen, da auch die Paketinstaller zusätzliche virtuelle User und Gruppen einrichten. Ich empfehle, das System bei der Einrichtung von Programmen mit solch datensensiblen Anforderungen grundsätzlich aktuell zu halten.

Also vor der Installation der eigentlichen Pakete zunächst die Systemaktualisierung:

```
root@raspi3:~
# apt update
# apt full-upgrade
# systemctl reboot
```

Und dann werden die für den Mailserver benötigten Pakete installiert:

```
root@raspi3:~
# apt install dovecot-core dovecot-imapd dovecot-lmtpd dovecot-sieve
# apt install postfix getmail
# apt install libsasl2-modules openssl
```

Für dovecot sind für meine Installation insgesamt 4 Pakete notwendig.

1. core, welches die Kernroutinen des Mailservers enthält

2. imapd, welches die Imap-Funktion des Mail-Servers zur Verfügung stellt, mit dem wir uns über unseren Thunderbird-Client verbinden

3. lmtpd, welches ein lokales Äquivalent zum smptd ist, also ebenfalls ein Mail-Transfer-Protokoll. Hier ermöglicht es aber, dass Emails von einem lokalen Anwender an einen anderen lokalen Anwender auch lokal zugestellt werden und NICHT den Weg über das Internet gehen. Solcherart versendete Emails bleiben also „im Hause“.

4. sieve, welches die Scriptsprache und die dazugehörigen Programme für die Abarbeitung von Filterregeln bei der Behandlung eingehender Emails beinhaltet.

Bei der Installation von Postfix beantworten wir alle Punkte im Abfrage-Dialog einfach mit Enter und bestätigen die Default-Werte. Die Inhalte sind hier an der Stelle nicht ganz so wichtig ist, da später sowieso eine eigene und völlig neue Konfigurationsdatei eingerichtet wird.

Libsasl wird für die Berechtigungsprüfung benötigt, ohne dieses Paket ist die folgende Fehlermeldung zu erwarten: SASL authentication failed; cannot authenticate to server

Opensssl wird für die Erstellung der eigenen Root-CA sowie der Server-Zertifikate benötigt.

Der folgenden Befehl listet nach der Installation die installierten Pakete auf:

```
# dpkg -l | egrep "mail|imap|sasl|openssl" -i
ii  dovecot-core          1:2.3.13  amd64  secure POP3/IMAP server - core files
ii  dovecot-imapd         1:2.3.13  amd64  secure POP3/IMAP server - IMAP daemon
ii  dovecot-lmtpd         1:2.3.13  amd64  secure POP3/IMAP server - LMTP server
ii  dovecot-sieve         1:2.3.13  amd64  secure POP3/IMAP server – Sieve...
ii  getmail6              6.14-1    all    mail retriever ...
ii  libsasl2-2:amd64      2.1.27    amd64  Cyrus SASL - authentication ...
ii  openssl               1.1.1n    amd64  Secure Sockets Layer toolkit ...
ii  postfix               3.5.13    amd64  High-performance mail transport agent
```

Die erste und wichtigste Frage lautet nun: Wo wollen wir überhaupt unsere Emails speichern? Wie soll das alles dergestalt organisiert werden, so das uns spätere Folgearbeiten nicht unnötig erschwert werden? Dabei sollten wir bedenken, dass wir die Daten besser nicht fragmentiert anlegen, also z.B. nicht in Verzeichnissen, die nicht gemeinsam unter einem führenden Verzeichnis angelegt sind. Denn das macht es uns später unnötig schwer, mit wenigen Befehlen für das ganze Mail-System auf einmal ein Backup zu erstellen. Und genau das sollten wir heute schon für später als regelmäßige Aufgabe einplanen.
Weil mein Mailserver anfangs auf einem Raspberry Pi lief, war auch die in vielen Anleitungen empfohlene Speicherung im Home-Directory eines virtuellen Users ausgeschlossen. Da der RPi nur eine 8GB-Micro-SDC eingesetzt hatte, musste ich mir darüber wirklich keine Gedanken machen. Und wenn ich die Daten sowieso extern speichern muss, bei mir auf einer SSD, warum soll ich dann überhaupt noch den Umweg über ein virtuelles Home-Dir gehen, wenn ich die notwendigen Verzeichnisstrukturen doch gleich am endgültigen und tatsächlichen Speicherort auf der SSD einrichten kann?

Ich habe für mein Mailsystem die folgende anwenderbezogene Verzeichnisstruktur eingerichtet:



Eine meiner wesentlichen Prämissen in der Gestaltung der Verzeichnisstruktur zur Speicherung der Daten war, dass ich die Postfächer (mehrere!) eines Users rigoros von allen anderen Usern getrennt haben will. In einer wahrscheinlich üblichen Installation, in der jeder User genau ein Postfach besitzt, liegen vermutlich alle Postfächer hintereinander und gleichrangig innerhalb eines Mailverzeichnisses. Technisch ist das natürlich kein Problem, aber ich habe mich dagegen entschieden.

Ich hatte zuvor auch einige Zeit darüber nachgedacht, die Mailstruktur direkt in den Server-Home-Dirs der User zu speichern, die bei mir anwender-individuell von der SSD ins jeweilige Homedir gebindet sind, aber ich habe auch diesen Gedanken dann ganz schnell wieder verworfen. Denn damit besteht wieder die nicht unerhebliche Gefahr der Kompromittierung der Daten bzw. der Datenstruktur. Nicht mal so sehr durch Schadsoftware, sondern eher unbeabsichtigt durch den User selber, der mit ein paar Klicks seiner Anwendungsprogramme alles löschen könnte. Ein unmittelbarer Zugriff durch den Anwender selber ist an dieser Stelle auch überhaupt nicht notwendig. Nach meiner Überzeugung ist es deshalb eine gute Lösung, diese Daten einerseits aus dem unmittelbaren Wirkungsbereich des Users herauszuhalten, so dass Verzeichnisse und Daten nicht im Bereich der Samba-Shares liegen, auf die der User selber mit normalen Anwendungsprogrammen zugreifen kann. Und andererseits trotzdem alles zentral an einer Stelle zu speichern, um Backups mit geringem Aufwand erstellen zu können.

An meinem Server ist für die tagesaktuellen Bewegungsdaten der Anwender eine 256 GB SSD angeschlossen. Diese SSD ist wegen der eher geringen Kapazität nicht als Massendatenspeicher eingesetzt, sondern tatsächlich nur für die aktuellen und täglich in Gebrauch befindlichen Anwender-Daten wie das Mailsystem, Office-Dokumente, persönliche Dateien, usw.

Zur Aufnahme des täglichen Backups der SSD, für Multimedia-Dateien oder langfristig aufbewahrte ‚andere' Dateien sind zwei weitere Festplatten mit hoher Kapazität angeschlossen. Der wirklich markante Vorteil dieser SSD ist die Geschwindigkeit, denn aus Stromspargründen schicke ich meine Festplatten jeweils nach wenigen Minuten in den Standby. Und das frühere Warten, bis die HD aus dem Standby erwacht ist, kann manchmal schon echt nervend sein. Dieses Problem hat die SSD komplett beseitigt, ohne den Nachteil des ständig hohen Stromverbrauchs permanent laufender Festplatten zu haben, die möglicherweise auch noch einem hohen Verschleiß wegen ständig anlaufender Schrittmotoren unterliegen, weil sie sehr oft bei Pausen in den Ruhemodus wechseln.

Verschaffen wir uns als erstes einen kurzen Überblick über die verwendeten bestehenden Verzeichnisse, über weitere zum Teil noch einzurichtende Verzeichnisse und wir überlegen die weiteren Schritte:

Verzeichnis	Inhalt
/etc	Das Verzeichnis enthält die wichtigen Dateien „hosts, fstab, mailname“, die angepasst werden.
/etc/dovecot	...enthält die Konfigurationseinstellungen des Mailservers sowie die Mailserver-Anmeldedaten der User.
/etc/postfix	...enthält die Konfigurationseinstellungen des Mail-Transfer-Agents, dann die Anmeldedaten zum SMTP-Mail-ISP, Anwender-bezogene Richtlinien, Berechtigungs- und Ersetzen-Filter, usw.
/media/SSD/Mail/ Getmail Imap Sieve	Das Server-Gespeicherte Datenverzeichnis des Mailservers enthält - die Zugangsdaten zum Abholen bei den Mail-ISP - die gesendeten und empfangenen Emails aller User - die Filterregeln für die Postfächer für eingehende Emails
/usr/local/bin	...enthält den Eventhandler, der Anwenderbezogen die Emails vom Mail-ISP abholt, wenn sich der Anwender an seinem Client-PC über sein Mail-Client-Programm (z.B. Thunderbird) am Mailserver anmeldet.

Ich beginne bei der vorbereitenden Einrichtung des Mailservers mit Benutzern und Gruppen, also dem Benutzer-Management. Auf meinem Server sind alle Familienmitglieder, die künftig den Mailserver nutzen wollen, in der Gruppe „*vmail*“ zusammengefasst. Der User „*vmail*“, dem diese Gruppe gehört, ist ein virtueller User, also keine reale Person. „*vmail*“ ist schließlich auch der einzige User, unter dessen UID die Zugriffe auf die Postfächer erfolgen. Das bedeutet, wenn sich ein realer Anwender anmeldet und seine Anmeldung war gültig, dann erfolgt der Zugriff auf die Postfächer dieses Anwender über einen Kontext-Wechsel auf den User „*vmail*“. Keine gültige Anmeldung = kein Kontextwechsel, kein Zugriff.

Die Mitgliedschaft der Mailserver-Anwender in dieser Gruppe ist notwendig, weil der Dovecot-Service gemäß der Einstellung in der Konfiguration nur auf Mitglieder dieser Gruppe beschränkt ist. Das bedeutet, nur die in dieser Gruppe enthaltenen Mitglieder sind zur Benutzung des Mailservers berechtigt.

Sofern der User „*vmail*“ mit seiner Gruppe noch nicht existiert, wird er mit

```
# useradd -u 5000 vmail
```

angelegt. Die UID 5000 ist willkürlich gewählt, damit sie sich deutlich von den UIDs der normalen Anwender abgrenzt. Die einzelnen Benutzer werden anschließend nacheinander mit

```
# adduser thomas vmail
# adduser silvia vmail
# adduser manuel vmail                  usw.
```

der Linux-Gruppe vmail hinzugefügt. Ich gehe an diesem Punkt angekommen davon aus, dass auf dem Linux-Server alle späteren Mailuser bereits mit Name und Password eingerichtet sind. Falls das nicht zutrifft, so müssen nun alle späteren Anwender ebenfalls angelegt werden und der Gruppe Mail hinzugefügt werden.

Darüber hinaus ist noch 1 weiterer fehlender „User“ anzulegen, der hier aber nur ein virtueller User und keine reale Person ist. Hierbei geht es um den User „familie“, der später das gemeinsam genutze Familien-Postfach bereitstellt. Bitte hier unbedingt das Passwort notieren!

```
# adduser familie
# adduser familie vmail
```

Und weil ich auch immer wieder mal von Samba-Shares spreche, auch wenn das nicht unbedingt zum aktuellen Thema passt, so möchte ich doch kurz erwähnen, mit welchem Befehl man sich mal eben anschauen kann, ob schon Mitglieder eine Sambaberechtigung haben und welche das sind:

Die drei folgenden Befehle zeigen uns zur Kontrolle die angelegten Linux-User, Mitglieder der Gruppe „vmail“ und die Samba-User:

```
# awk -F':'  '{ printf "%6s %-18s %10s\n\r", $3, $1, $4}' /etc/passwd | sort -b -g
# getent group vmail
# pdbedit -L
```

Nun wird das Speichermedium, die zuvor genannte SSD, für die Speicherung des späteren Mailverkehrs eingerichtet. Das kann natürlich auch eine beliebige andere angeschlossene oder interne Festplatte sein, ggf. sogar ein Netzwerkspeicher. Der Speicher muss nur gemountet werden und es muss sichergestellt sein, dass er immer verfügbar ist. Meine SSD wird in der „*fstab*“ gemountet:

```
UUID=8d12a6 /media/SSD ext4 rw,noexec,auto,nouser,async,noatime 0 0
```

Dann den Mountpoint einrichten, das Laufwerk mounten und kontrollieren, ob es erfolgreich war:

```
# mkdir /media/SSD
# mount -a
# df
```

Wegen der für das Verstehen notwendigen Kontinuität und der Einfachheit halber bleibe ich hier konsequent bei der Bezeichnung „SSD“. Wenn ein anderer Speicher genutzt wird, muss halt die Bezeichnung angepasst werden. Wenn der andere Speicher korrekt eingerichtet ist, hat das keine störende Auswirkung auf den weiteren Verlauf dieser Anleitung.

Die folgenden Postfächer sollen unter Dovecot für die Anwender eingerichtet werden:

Dovecot Postfachname	**ISP**	**Externe Mailadresse**
thomas.addams_t	toml.de	thomas.addams@toml.de
thomas.addams_g	gmx.de	thomas.addams@gmx.de
tom.addams	gmail.com	tom.addams@gmail.com
silvia.addams_t	toml.de	silvia.addams@toml.de
silvia.addams_g	gmx.de	silvia.addams@gmx.de
manuel.addams_t	toml.de	manuel.addams@toml.de
manuel.addams_g	gmx.de	manuel.addams@gmx.de
familie	toml.de	thomas+silvia.addams@_toml.de

Folgende neue Verzeichnisse sind dazu auf der SSD einzurichten:

```
mkdir /media/SSD/Mail
mkdir /media/SSD/Mail/Getmail
mkdir /media/SSD/Mail/Imap
mkdir /media/SSD/Mail/Sieve
```

Diese 3 Verzeichnisse beginnen bei mir als Zeichen dafür, dass es keine User sind, mit einem Großbuchstaben.

```
mkdir /media/SSD/Mail/Getmail/familie
mkdir /media/SSD/Mail/Getmail/thomas
mkdir /media/SSD/Mail/Getmail/silvia
mkdir /media/SSD/Mail/Getmail/manuel
usw.

mkdir /media/SSD/Mail/Imap/familie
mkdir /media/SSD/Mail/Imap/thomas
mkdir /media/SSD/Mail/Imap/silvia
mkdir /media/SSD/Mail/Imap/manuel
usw.
```

Und weiterhin, für jeden User ein individuelles Verzeichniss.

Die individuellen usereigenen Imap-Ordner sind nicht zwingend erforderlich. Es ist auch möglich die Postfächer linear untereinander in einem Verzeichnis zu speichern. Dazu muss aber die dovecot.conf und die passwd angepasst werden. Aus Gründen der Übersichtlichkeit mag ich es aber lieber getrennt.

```
mkdir /media/SSD/Mail/Sieve/familie
mkdir /media/SSD/Mail/Sieve/thomas.addams_t
mkdir /media/SSD/Mail/Sieve/thomas.addams_g
mkdir /media/SSD/Mail/Sieve/silvia.addams_t
mkdir /media/SSD/Mail/Sieve/silvia.addams_g
mkdir /media/SSD/Mail/Sieve/manuel.addams_t
mkdir /media/SSD/Mail/Sieve/manuel.addams_g
usw.
```

Die Sieve-Verzeichnisse enthalten das Dovecot-Homedir für jedes Postfach, dessen Name identisch mit dem Dovecot-Postfach bzw. dem Namen des virt. Dovecot-Users ist.

Für die zwei Verzeichnisse Getmail und Imap werden die tatsächlichen Anwendernamen verwendet, also die Linux-Namen, mit denen sich die Anwender z.B. auf ihrem PC oder auch für die Samba-Freigaben anmelden. Im Verzeichnis Sieve werden die virtuellen Anwendernamen des Dovecot-Mailservers verwendet, mit anderen Worten die lokalen Postfachnamen der Mail-Anwender.

Als nächstes werden die Rechte auf diese neuen Verzeichnisse entsprechend der folgenden Tabelle gesetzt:

Dir	User	Gruppe	Alle	Rechte		Verzeichnis
d	rwx	r-x	r-x	755	root:root	/media/SSD
d	rwx	rwx	---	770	vmail:vmail	/media/SSD/Mail
d	rwx	rwx	---	770	vmail:vmail	/media/SSD/Mail/Getmail
d	rwx	rwx	---	770	vmail:vmail	/media/SSD/Mail/Imap
d	rwx	rwx	---	770	vmail:vmail	/media/SSD/Mail/Sieve

```
# chmod 755 /media/SSD
# find /media/SSD/Mail/ -type d -exec chmod 770 {} +
# find /media/SSD/Mail/ -exec chown vmail:vmail {} +
```

Die Besonderheit bei Sieve ist der Precompiler, mit dem die Sievescripte zur Laufzeit des Programms, und während der Zustellung kompiliert werden. Das geschieht u.U. beim Abholen der EMails durch Getmail und beim Anschließenden durchlaufen der Emails durch Sieve, bevor sie ins Postfach des Users zugestellt werden. Der Job läuft unter der UID des Users, also ist hier auf das Verzeichnis eine +W-Berechtigung für die Gruppe "mail" erforderlich.

Die Rechte für die Dateien in den beiden Verzeichnissen Sieve und Getmail werden wir später manuell setzen, nachdem wir die dort anzulegenden Dateien erstellt haben. Im Verzeichnis Imap werden die Rechte beim Speichern der Emails automatisch passend durch dovecot gesetzt.

Die benutzereigenen Unterverzeichnisse erhalten allesamt einheitliche Userrechte, entsprechend dem hier folgenden Beispiel des Users „thomas“ auf ‚seine‘ Verzeichnisse. In gleicher Manier sind die Rechte individuell für die Unterverzeichnisse aller anderen Dovecot-User zu setzen:

Dir	User	Gruppe	Alle	Rechte		Verzeichnis
d	rwx	rwx	---	770	vmail:vmail	/media/SSD/Mail/Getmail/thomas
d	rwx	rwx	---	770	vmail:vmail	/media/SSD/Mail/Imap/thomas
d	rwx	rwx	---	770	vmail:vmail	/media/SSD/Mail/Sieve/$tomspostboxes

Mit dem folgenden Befehl schauen wir uns nun das Ergebnis unserer Einrichtung an und prüfen, ob alle Verzeichnisse mit den korrekten Angaben gesetzt sind:

```
# find /media/SSD/Mail/ -type d -ls
```

Mit den folgenden Befehlen werden wir später nach vollständiger Einrichtung erneut einmalig die Rechte neuer Dateien anpassen:

Allen Verzeichnissen 770 [rwx rwx ---] zuweisen:

```
# find /media/SSD/Mail/ -type d -exec chmod 770 {} +
```

Allen Dateien in den Verzeichnissen 660 [rw- rw- ---] zuweisen:

```
# find /media/SSD/Mail/ -type f -exec chmod 660 {} +
```

Allen Dateien und Verzeichnissen die Besitzrechte zuweisen:

```
# find /media/SSD/Mail/ -exec chown vmail:vmail {} +
```

Damit sind die vorbereitenden Arbeiten hinsichtlich Linux-User für Datei- und Verzeichnisberechtigungen abgeschlossen und es geht weiter mit einem sicherheitstechnischen Kapitel und dann der Einrichtung des eigentlichen Mailservers „dovecot“ und des Mail-Transfer-Agents „postfix“.

8. Openssl - Datensicherheit mit TLS - Self-Signed Certificates

Mit diesem jetzt eingefügten neuen Kapitel pack ich nun -um es mal sprichwörtlich auszudrücken- ein echt heißes Eisen an. Auch dieses Thema enthält mal wieder ein großes Potential für Missverständnisse und Fehlinterpretationen … es ist ein wirklich anspruchsvolles Thema! Openssl ist (nach wochenlanger Recherche für Erklärungen, Tutorials, Zusammenhänge, Lösungen, Test-Server-Betrieb und vieles weitere mehr) in meinem Fazit nicht einfach nur ein Programm, was man mal eben zur Verbesserung der Datensicherheit beim Transport als Paket aus dem Repository installiert und startet. Openssl ist viel mehr ein Sicherheitskonzept und i.ü.S. ein wenig auch eine digitale Wissenschaft, die einiges über das hinausgeht, was ich mal eben auf die Schnelle als Laie autodidaktisch verinnerlichen kann. Der Aufwand und die einzelnen Schritte zur TLS-Implementierung in den Anwendungen ist zudem fast immer auch anwendungsspezifisch, was das Verstehen der Zusammenhänge ebenfalls nicht gerade vereinfacht. Die Möglichkeiten eigene Zertifikate zu erstellen sind bestenfalls mit der Metapher *„Viele Wege führen nach Rom“* umschrieben. Insofern *„fahre ich zwar jetzt das Auto auf dem Weg nach Rom, verlass mich aber auch zu einem großen Teil darauf, dass mich die Assistenzsysteme vor einem Unfall bewahren“*.

In der Vorversion meines Setups hatte ich mich noch damit begnügt, einfache Zertifikate für die Verschlüsselung zu verwenden, die ich mit dem folgenden Befehl erzeugt habe:

```
$ openssl req -new -x509 -days 3650 -nodes -out "/tmp/dovecot.cert" -keyout "/tmp/dovecot.key" -outform PEM
```

Die Befehlszeile oberhalb ist **ein** zusammenhängender langer Befehl, der auch unbedingt unberücksichtigt eines evtl. angezeigten Zeilenumbruchs als ganzes und komplett abgesetzt werden musste!

Das von mir erstellte und bei einem Connect-Versuch durch ein Client-System vom Server an den Client übertragene Server-Zertifikat enthält zum einen die „*Behauptung*“, wer der Ersteller dieses Zertifikats ist, und zum anderen den für einen verschlüsselten Datentransfer im lokalen Netz notwendigen Public-Key zur Ver- und Entschlüsselung des Traffics durch das Client-System. Der dazu passende Private-Key für die gleiche Aufgabe liegt auf dem Server. Ich habe die Wortwahl „*Behauptung*“ bewusst getroffen, denn wenn es so einfach ist, ein Zertifikat mit diesen Angaben zu erzeugen, so kann das durchaus auch ein möglicher Angreifer tun. Ein potentielles Problem entsteht also genau in dem Moment, wenn es einem Angreifer mit zweifelhaften Absichten gelungen ist, ein Fake-Zertifikat mit eben diesen Angaben auf dem Rechner zu installieren, welches dann behauptet, von mir erstellt worden zu sein, aber vielleicht einen ganz anderen Server (MITM irgendwo im Internet) legitimiert.

Ich habe mich nun deshalb dazu entschieden, dieses einfache Zertifikat durch ein signiertes Zertifikat zu ersetzen und zusätzlich den Datenverkehr zwischen Mail-Client <> Dovecot-Server von STARTTLS auf TLS/SSL umzustellen. Da es sich bei meinem Server bis auf die ausgehenden Emails um einen Server ohne Zugang von außen handelt, der also nicht über das Internet erreicht werden kann, kann man diesen Punkt sicher diskutieren und ggf. überlegen, darauf zu verzichten. Aber vor dem Hintergrund, dass bei uns auch potentiell hochgradig unsichere Systeme mit Windows und Android laufen, über die ich faktisch keine oder allenfalls eine illusorische Kontrolle über deren Betriebssysteme und proprietäre Anwendungen habe, möchte ich doch den lokalen Datenverkehr nach besten Wissen und Gewissen sichern. So sind z.B. jetzt schon auch unsichere Samba-Verbindungen über Netbios (nmbd) untersagt und es wird dafür ebenfalls TLS verwendet.

Was bedeutet nun „signiertes Zertifikat“? Das ist einfach, man wendet sich für sein eigenes Zertifikat mit seiner Zertifikatsanfrage (CSR=Certificate Signing Request) an eine offizielle Zertifizierungsstelle. Das sind bestimmte Dienstleistungsunternehmen, wie z.B. Let’s Enrypt, Cloudflare, Startssl.com o.ä. Unternehmen, die dann das von mir selber erstellte Zertifikat mit einem eigenen Key-Identifier signieren und damit bestätigen, dass die Angaben dieses Zertifikats der Wahrheit entsprechen. Bevor mein Zertifikat allerdings signiert wird, prüft diese Authentifizierungs-Stelle natürlich alle meine Angaben auf Wahrheitsgehalt… ein Aufwand, der von einigen Unternehmen dann durchaus auch in Rechnung gestellt wird. Es kann sein, dass man den Personalausweis, oder den Reisepass, oder den Führerschein, vielleicht zusammen mit den letzten Telefonabrechnungen digitalisiert einsenden muss. In der Regel sichern sich diese Signing-Dienste mit einer zeitlich limitierten Gewährleistung der von ihnen *„bestätigten Wahrheit der Angaben“* damit ab, dass ein solches signiertes Zertifikat eben nur begrenzt gültig ist. Das bedeutet in der Folge, dass es (wieder gegen Berechnung) periodisch automatisch erneut geprüft und bestätigt werden muss, weil es andernfalls durch Ablauf des Gültigkeitszeitraums ungültig wird.

Dabei kann man durchaus feststellen, je seriöser und sorgfältiger die Überprüfung der von mir behaupteten „digitalen Identität“ auf dem Zertifikat ist, umso besser ist nachher die Chance eine sichere Verbindung mit dem erwarteten (!) Server auch zu gewährleisten. Es ist aber nicht zwingend notwendig, sich für den rein privaten Server-Betrieb für signierte Zertifikate an einen offiziellen Dienstleister zu wenden. Man kann sich durchaus auch eine eigene Signing-Authority, eine Root-Certificate-Authority erstellen. Mit einem nicht-öffentlichen Root-Zertifikat zusammen mit dem geheimen Root-Schlüssel, die ich beide getrennt und sicher vom allgemeinem Netzwerkbetrieb aufbewahre, kann ich dann meine Server-Zertifikate selber signieren, z.B. auch für diesen Mailserver

An dieser Stelle komme ich noch einmal auf „Behauptung“ zurück und zeige nun, wie man das im Fall der Fälle bei einer Misstrauens- oder Fehlervermutung kontrollieren kann. Es gibt fast immer in den Anwendungsprogrammen (wie Firefox oder Thunderbird) die Möglichkeit, sich die installierten Zertifikate anzusehen. Und genau damit hab ich auch die Möglichkeit, die Key-Identifier des Client-Zertifikats mit den tatsächlichen auf meinem Server zu vergleichen. Im Vergleich zu dem oben zuerst genannten einfachen Zertifikat, in dem die Werte für Subject und Authority immer gleich sind, ist bei einem mit einer Root-CA signierten Zertifikat neben dem Server-Identifier (Subject) auch der Root-Authority-Identifier hinterlegt. Und wenn der auf Client-Seite nicht mit dem meiner Root-CA auf dem Server hinterlegten Zertifikat übereinstimmt, würde ich annehmen, da ist was faul im System. Soweit ich das bisher verstanden habe, ist es nicht möglich, in ein Fake-Zertifikat mal eben diese Root-Key-ID reinzukopieren, zumal man weiterhin davon ausgehen muss, dass dem Angreifer weder die Root-Key-ID bekannt ist und ihm sowieso meine Root-CA fehlt, um ein eigenes Fake-Cert damit zu signieren… was alles zusammengenommen schließlich ein nachvollziehbarer Grund für die sichere Aufbewahrung meiner eigenen Root-CA ist.

Später, wenn Dovecot und Postfix eingerichtet sind und laufen, kann ich mir *(hier als Beispiel von einem Linux-)* Client-System die echten Daten anzeigen lassen und diese ggf. mit den Angaben eines Client-Certs vergleichen. „mail’ ist hier der DNS-Name meines Mail-Servers im lokalen Netz, an der Stelle kann auch die lokale IP-Adresse des Servers eingetragen werden.

```
$  echo "Q" | openssl s_client -connect mail:993 2>1 | openssl x509 -text -noout | grep -A1 'Key Identifier'
        X509v3 Subject Key Identifier:
                52:62:B3:0A:58:3F:49:E0:45:A0:85:B7:45:9D:29:85:1D:FA:3C:58
        X509v3 Authority Key Identifier:
                keyid:6B:90:85:F3:4E:56:5E:3B:13:36:DA:D4:A9:01:FA:67:53:EF:FC:4F
```

Das gleiche funktioniert mit Postfix, wobei ich allerdings hierfür die gleichen Zertifikate wie für Dovecot verwende, weswegen auch das Ergebnis identisch wäre:

```
$  echo "Q" | openssl s_client -connect mail:25 -starttls smtp 2>1 | openssl x509 -text -noout | grep -A1 'Key Identifier'
```

Soweit also ein paar Grundlagen vorab, nun befasse ich mich mit dem Erstellen meiner neuen signierten Zertifikate in 5 Schritten. Wir beginnen mit der Sicherung der bestehenden openssl.cnf. Hierzu ist anzumerken, dass wir die neuen Zertifikate natürlich auf jedem beliebigen Linux-Client-Rechner erstellen könnten. Ich tue das aber auf dem Server via SSH, weil ich davon ausgehe bzw. mit hoher Wahrscheinlichkeit sogar ausschließen kann, dass mein Server kompromittiert ist.

```
root@raspi3:/etc/ssl
# cp /etc/ssl/openssl.cnf /etc/ssl/openssl.cnf.org
```

Anschließend ändern wir in der bestehenden openssl.cnf nun zwei Parameter in der Sektion [CA_default], und zwar

```
#dir            = ./demoCA           # Where everything is kept
dir             = /tmp/pki/CA
default_days    = 3650
```

Ich habe mich für eine Laufzeit von 10 Jahren für meine Zertifikate entschieden. Man kann das sicher kürzer wählen, aber aufgrund der insgesamt isolierten Installation unseres Systems bewerte ich 10 Jahre hier als unkritisch, zumal ich davon ausgehen, dass ich beim nächsten Releasewechsel sowieso alle Zertifikate neu erstelle, wie z.B. auch für OpenVPN. Die PKI wird in /tmp erzeugt und anschließend einfach gelöscht, weil ich mit dieser Root-CA keine neuen Zertifikate zu erzeugen beabsichtige.

1. Vorbereiten der für die PKI notwendigen Infrastruktur.

Ich habe mich dazu entschieden, die komplette Aktion in /tmp durchzuführen, um sicher zu sein, dass nach dem Ausschalten des Systems alle angelegten Dateien gelöscht werden. Das erfordert natürlich, die neu erstellten Zertifikate vorher zu sichern.

```
$ mkdir -p /tmp/pki/CA/private; mkdir -p /tmp/pki/CA/newcerts

$ echo "00" > /tmp/pki/CA/serial
$ touch /tmp/pki/CA/index.txt

$ cd /tmp/pki/CA
$ ls
```

2. Manuelle Erstellung einer eigenen Certificate Authority (CA)

Diese CA als ‚prüfende und signierende Autorität' bestätigt die Echtheit der "Person", die auf dem späteren Zertifikat angegeben ist. Lapidar könnte man sagen, ich bestätige damit, dass ich ich bin und das/mein späteres Zertifikat von mir als zeichnende Autorität signiert wurde.

Die unter Punkt 4 folgende Signierung erwartet das Key-File im Verzeichnis ./CA/private/, wo es aber nicht angelegt wurde, deswegen legen wir einfach einen Link dahin. Ich empfehle sowieso, die Namen der Files so zu belassen, weil gewisse Prozesse über eine anscheinende Namenskonvention teilweise eben genau diese Namen erwarten. Behält man das bei, erspart man sich auf jeden Fall mögliche Probleme und Fehlermeldungen.

```
$ openssl req -new -x509 -newkey rsa:4096 -keyout cakey.pem -out cacert.pem -days 30
    Generating a RSA private key
    ..................................++++
    writing new private key to 'cakey.pem'
    Enter PEM pass phrase: GeheimesPWD
    Verifying - Enter PEM pass phrase: GeheimesPWD
    -----
    You are about to be asked to enter information that will be incorporated
    into your certificate request.
    What you are about to enter is what is called a Distinguished Name or a DN.
    There are quite a few fields but you can leave some blank
    For some fields there will be a default value,
    If you enter '.', the field will be left blank.
    -----
    Country Name (2 letter code) [AU]:DE
    State or Province Name (full name) [Some-State]:MeinBundesland
    Locality Name (eg, city) []:MeineStadt
    Organization Name (eg, company) [Internet Widgits Pty Ltd]:toml
    Organizational Unit Name (eg, section) []:.
    Common Name (e.g. server FQDN or YOUR name) []:toml.de
    Email Address []:admin@toml.de

$ ln -s /tmp/pki/CA/cakey.pem /tmp/pki/CA/private/cakey.pem
```

Es ist nun möglich, ein paar Prüfungen vorzunehmen und sich die Ergebnisse anzeigen zu lassen:

```
$ openssl x509 -in cacert.pem -text -noout
$ openssl x509 -enddate -noout -in cacert.pem
$ openssl rsa -noout -text -in cakey.pem
```

3. Einen privaten Schlüssel für das Serverzertifikat und die Diffie-Hellman-Datei für den Key-Exchange erzeugen

Bei der Erstellung des Keyfiles wird die Eingabe einer Passphrase erwartet. Nachdem das Keyfile erzeugt wurde, entfernen wir diese Passphrase wieder, weil sie nicht benötigt wird und ggf. sogar kontraproduktiv sein kann. Beim Start des Servers/Service wäre es wenig wünschenswert, wenn während des Starts dieses Services auf die Eingabe eines Passwords gewartet werden würde.

```
$ openssl genrsa -out serverkey.pem -aes256 4096
$ openssl dhparam -out dh4096.pem 4096

$ openssl rsa -in serverkey.pem -out serverkey_2.pem
$ mv serverkey_2.pem serverkey.pem -f
```

4. Ein Certificate Signing Request mithilfe des vorher erzeugten privaten Server-Keys erzeugen, um anschließend damit ein öffentliches digitales und signiertes Zertifikat mit dem öffentlichen Pubkey zu erstellen.
Hier unbedingt den Unterschied (vergl. Pkt. 2) beim Parameter „Common Name“ beachten, hier wird die lokale Mail-Domain eingetragen.

```
$ openssl req -new -key serverkey.pem -out csr.pem -nodes
    You are about to be asked to enter information that will be incorporated
    into your certificate request.
    What you are about to enter is what is called a Distinguished Name or a DN.
    There are quite a few fields but you can leave some blank
    For some fields there will be a default value,
    If you enter '.', the field will be left blank.
    -----
    Country Name (2 letter code) [AU]:DE
    State or Province Name (full name) [Some-State]:MeinBundesland
    Locality Name (eg, city) []:MeineStadt
    Organization Name (eg, company) [Internet Widgits Pty Ltd]:toml
    Organizational Unit Name (eg, section) []:.
    Common Name (e.g. server FQDN or YOUR name) []:mail
    Email Address []:toml@mail

    Please enter the following 'extra' attributes
    to be sent with your certificate request
    A challenge password []:
    An optional company name []:

$ openssl req -noout -text -in csr.pem
```

5. Mit der eigenen CA (vergl. Pkt.1) das Server-Zertifikat erstellen und signieren. Damit wird die Echtheit des Servers bestätigt. Darüber hinaus enthält es auch den von den Clients verwendeten Public-Key.

```
$ openssl ca -in csr.pem -notext -out servercert.pem
$ openssl x509 -in servercert.pem -text -noout
```

Abschließend wird noch ein Chain-File für die Übernahme in die dovecot.conf erzeugt und die Rechte gesetzt …

```
$ cat servercert.pem cacert.pem > server_ca_chain-cert.pem

$ su -
# chmod 400 *.pem
# chown root:root *.pem

# ls
```

… und zum Schluss die alte openssl.cnf im Verzeichnis /etc/ssl wiederherstellen.

Sofern alles korrekt abgelaufen ist, finden wir nun u.a. die folgenden Dateien im Verzeichnis, welches genau die Dateien sind, die wir für Postfix und Dovecot verwenden und deshalb vor dem Löschen des temporären Verzeichnisses in /tmp sichern müssen.

```
root@raspi3:/
# ls /tmp/pki/CA
drwxr-x--- 2 root root    4,0K 2022-07-19 15:04 .
drwxr-xr-x 7 root root    4,0K 2022-06-24 15:43 ..
-r-------- 1 root root    2,0K 2022-06-24 15:21 cacert.pem
-r-------- 1 root root     769 2021-02-27 18:00 dh4096.pem
-r-------- 1 root root    4,0K 2022-07-19 15:04 server_ca_chain-cert.pem
-r-------- 1 root root    2,0K 2022-06-24 15:22 servercert.pem
-r-------- 1 root root    3,2K 2022-06-24 15:22 serverkey.pem
```

```
# mkdir -p /etc/dovecot/certs
# mkdir -p /etc/postfix/certs
```

```
# cp /tmp/pki/CA/server_ca_chain-cert.pem  /etc/dovecot/certs
# cp /tmp/pki/CA/serverkey.pem             /etc/dovecot/certs
# cp /tmp/pki/CA/dh4096.pem                /etc/dovecot/certs

# cp /tmp/pki/CA/cacert.pem                /etc/postfix/certs
# cp /tmp/pki/CA/servercert.pem            /etc/postfix/certs
# cp /tmp/pki/CA/serverkey.pem             /etc/postfix/certs
# cp /tmp/pki/CA/dh4096.pem                /etc/postfix/certs
```

```
# ls -lah /etc/dovecot/certs /etc/postfix/certs
/etc/dovecot/certs:
insgesamt 20K
drwxr-x--- 2 root dovecot 4,0K 2022-07-19 16:40 .
drwxr-xr-x 7 root root    4,0K 2022-06-24 15:43 ..
-r-------- 1 root root     769 2021-02-27 18:00 dh4096.pem
-r-------- 1 root root    4,0K 2022-07-19 15:04 server_ca_chain-cert.pem
-r-------- 1 root root    3,2K 2022-06-24 15:22 serverkey.pem

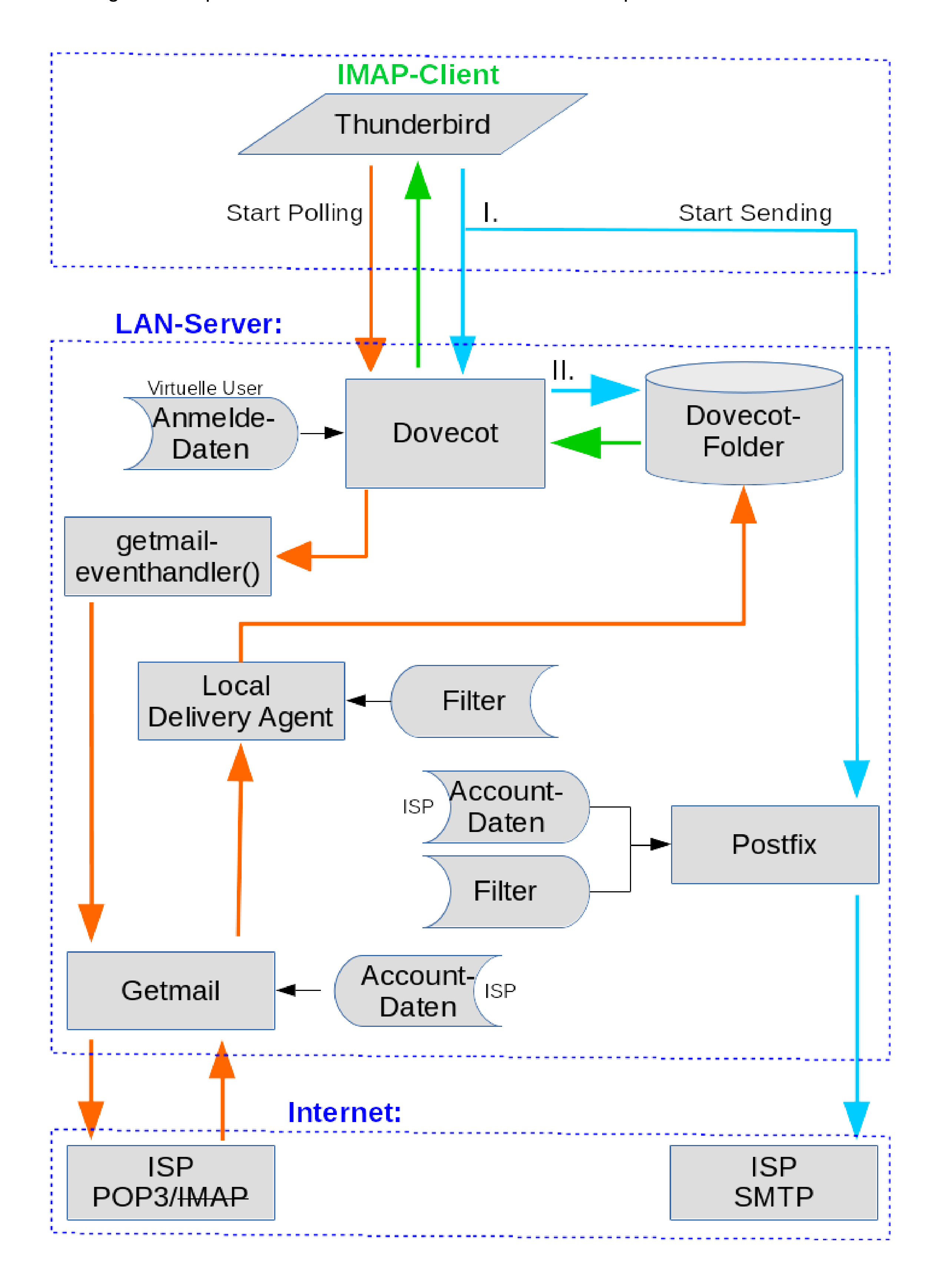
/etc/postfix/certs:
insgesamt 24K
drwxr-x--- 2 root postfix 4,0K 2022-07-19 16:39 .
drwxr-xr-x 8 root root    4,0K 2022-07-19 16:30 ..
-r-------- 1 root root     769 2021-02-27 18:00 dh4096.pem
-r-------- 1 root root    2,0K 2022-06-24 15:21 cacert.pem
-r-------- 1 root root    2,0K 2022-06-24 15:22 servercert.pem
-r-------- 1 root root    3,2K 2022-06-24 15:22 serverkey.pem
```

Später, bei der ersten Anmeldung mit Thunderbird an unseren Mailserver, wird uns Thunderbird fragen, ob das Zertifikat akzeptiert werden soll. Damit ist dann neben der Verschlüsselung -nach meinem Kenntnisstand- auch sichergestellt, dass unsere Anmeldung anschließend nicht auf einen anderen Server ohne unser Wissen umgeleitet werden kann. Es ist halt eine Sicherungsmaßnahme aus dem großen Paket der möglichen Sicherungsmaßnahmen. Würde zu einem späteren Zeitpunkt erneut die Zertifikat-Abfrage kommen, können wir sicher sein, dass sie nicht von unserem Server kommt. Vielleicht steht ein Versuch dahinter, unseren Datenverkehr umzuleiten oder zu belauschen. In diesem Fall sollte man besser sehr aufmerksam sein und genau überlegen, ob und was ggf. zu tun ist.

9. MDA und MTA – Workflow und Regelwerk

Dovecot => MDA => Mail Delivery Agent => Zustellung von Emails in Postfächer
Postfix => MTA => Mail Transfer Agent => das tatsächliche Senden der EMail

Bevor wir jetzt mit der Einrichtung der eigentlichen Programme beginnen, werfen wir noch mal kurz einen Blick auf den Workflow, um zu verstehen, wann und in welcher Richtung die Daten – also unsere Emails – künftig überhaupt durchs Netz fließen und was wir überhaupt zu erreichen versuchen:



Dabei möchte ich ein wenig das Augenmerk auf diese Symbole lenken, denn diese Symbole zeigen, an welchen Stellen wir jetzt manuell tätig werden müssen. Hier erwartet uns erneut neben den primären Konfigurations-einstellungen der Programme und dem bisher schon aufwendig gewesenen Benutzer-Management ebenfalls ein einigermaßen großer Arbeitsaufwand mit viel Detail-Kram. Es handelt sich in der Summe dieser Schritte in gewisser Weise um ein Regelwerk, welches Erlaubnisse erteilt, oder bestimmte Aspekte verbietet, oder bei Abzweigungen flexibel die Richtung vorgibt, oder Zuordnungen von Subjekten (Usern) und Objekten (Postfächern) festlegt. Wir müssen hier sowohl die ISP-Account-Daten als auch die Zugangsdaten der Anwender zu unserem Server hinterlegen, denn ohne Anmeldedaten geht gar nichts.

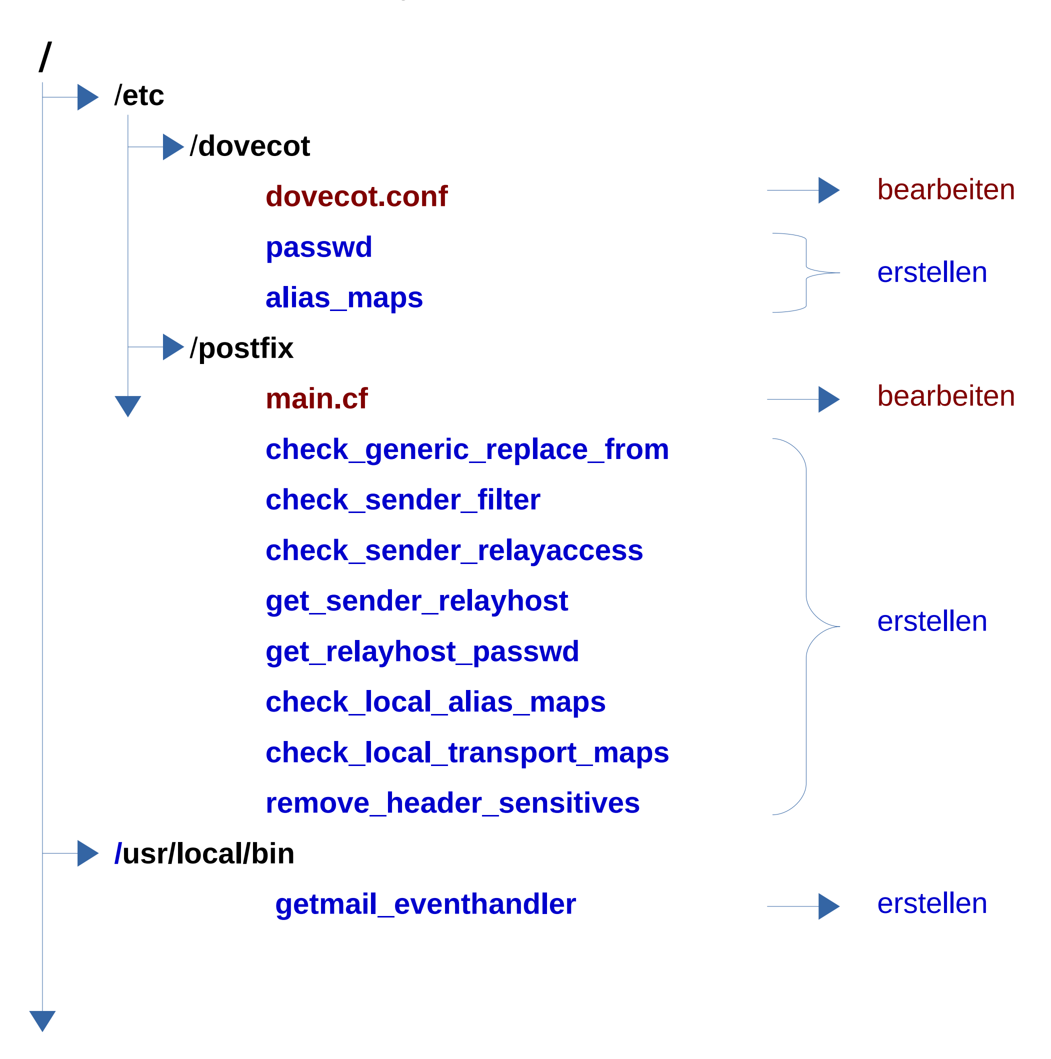


Es müssen weiterhin mehrere Filter mit unterschiedlichsten Verantwortlichkeiten eingerichtet werden:

- Wer darf mailen?
- Wie wird die Post an die richtigen Empfänger verteilt?
- Welcher User ist welchem Mail-ISP zuzuordnen?
- Wie wird interne Post (innerhalb des LANs ohne Internet) behandelt?

Dieses Regelwerk wird zusätzlich neben den zu den Programmen gehörenden Konfigurationsdateien (*.confs) in mehreren zusätzlichen Dateien hinterlegt, die jede Datei für sich einen ganz besonderen Zweck erfüllt.

Hier folgt nun ein Überblick über die jetzt betroffenen Verzeichnisse und Dateien, die zur Einrichtung bearbeitet werden müssen. Dateien, die letztendlich das Regelwerk beinhalten und beschreiben, mit denen wesentliche Abläufe gesteuert und kontrolliert werden:



Die jetzt folgenden Aufgaben, die Erstellung des Regelwerks, muten ganz sicher ein wenig wie *"boar, ist das viel und kompliziert"* an… das liegt ganz einfach an der Menge der notwendigen und betroffenen Dateien für dieses Regelwerk. Aber ich werde versuchen, das hier gradlinig und kontinuierlich von oben nach unten darzustellen und natürlich auch die wichtigen Zusammenhänge zu erklären.

Neben den eigentlich Konfigurationsdateien der Programme sind es diese Dateien, die wir manuell anlegen bzw. bearbeiten müssen:

	Datei:	Bedeutung:
/etc/dovecot/	passwd	Enthält die verschlüsselten Zugangsdaten der virtuellen User zum dovecot-Mailserver
▼	alias_maps	Die Zuordnung eines virtuellen Users zum Linux-User
/usr/local/bin	getmail_eventhandler	Abruf der Emails nach Start des Handlers durch das Dovecot-Post-Login-Scripting
/etc/postfix/	check_generic_replace_from	Übersetzt die lokale Mailadresse in die internettaugliche FQDN-Emailadresse des Mail-ISP FQDN = Fully qualified domain name Mail-ISP = Mail – Internet-Service-Provider
	check_sender_filter check_sender_relayaccess	Festlegen, welche lokalen Absender-Email-Adressen zum Senden von Emails berechtigt sind und welche Sender einen Relay-Host verwenden dürfen, oder andersrum, wem es verboten ist zu senden
	check_local_alias_maps check_local_transport_maps	Steuert die Verwendung lokaler Email-Empfänger. Damit wird erreicht, dass Emails an lokale Empfänger nicht durchs Internet versendet werden, sondern direkt im Postfach des lokalen Empfängers als Posteingang gespeichert werden
	get_sender_relayhost	Zuordnung des Relay-Host (SMTP-Server des Mail-ISP) zu den lokalen Emailadressen
	get_relayhost_passwd	Zuordnung der Anmeldedaten für den Relayhost, resp. für das zur lokalen Emailadresse passenden Postfach beim Mail-ISP
▼	remove_header_sensitives	Entfernt einige sensible lokale Informationen aus dem Mailheader, die weder für den Empfänger noch für den Sendevorgang an sich eine Bedeutung haben



Alle hier in dieser Beschreibung angelegten bzw. anzulegenden Dateien können mit dem dieser Beschreibung entsprechenden Inhalt unter der folgenden Adresse runtergeladen werden. Wenn die im Tar-File enthaltenen Dateien in die späteren Original-Verzeichnisse kopiert werden, um sie dort manuell anzupassen, müssen unbedingt die Rechte-Einstellungen gemäß dieser Anleitung nachbearbeitet werden! Im Übrigen ist es nicht unmöglich, dass die Beispiele hier in der Anleitung weniger aktuell sind, als die im Tar-File enthaltenen Dateien.

Deswegen empfehle ich unbedingt, nur die Beispieldateien aus dem Tar-File zu verwenden:

http://www.thlu.de/Public/TomsMailserver.tar

10. Dovecot – Einrichtung des Mail-Servers

Für alle jetzt folgenden Tätigkeiten empfehle ich noch die Installation zweier weitere Programme, die zwar nichts mit der Mailserver-Installation zu tun haben, die aber die Einrichtung und die spätere Betreuung unseres Servers deutlich erleichtern. Als erstes den „midnight commander“, der als einfacher Dateimanager im Terminal großes leistet und eine wirkliche Hilfe ist. Und als zweites mit dem „nice editor“ einen fast intuitiv bedienbaren Terminal-Texteditor. Natürlich funktionieren auch die altbekannten Editoren *nano* oder *vim*. Aber wer es halt ein wenig angelehnt an das in GUI-Editoren obligatorische und etablierte Handling haben möchte, ist mit dem „ne“ wirklich bestens bedient. Darüber hinaus bietet er die Möglichkeit, Tastaturbefehle und Shortcuts nach eigenem Ermessen einzustellen. Seit ich den „nice editor“ kenne, installiere ich nano und vim nicht mal mehr. Aber wie gesagt, das ist nur eine Empfehlung für diejenigen, die nicht ständiges Arbeiten im Terminal-Fenster gewohnt sind und sich den Einstieg nicht noch zusätzlich erschweren möchten.

```
# apt install mc ne
```

Wenn man beide Programme nicht kennt, würde ich jetzt vorher an dieser Stelle einige kleine Übungen einfügen, mal ein bisschen probieren und testen, um mich mit dem Umgang vertraut zu machen. Für den „nice editor“ existiert auf der Projekt-Seite im Web eine Dokumentation und Bedienungsanleitung.

Ebenfalls vorbereitend für die spätere Aufgabe zur Abholung unserer Emails vom ISP mit Getmail wird jetzt eine Datei angelegt, um ein Alias-Mapping vom virtuellen Dovecot-User zum regulären Linux-User zu ermöglichen. Dieses Mapping ist notwendig, damit die Abholung der Emails unter der eigenen Linux-User-ID des betroffenen Users erfolgen kann. Innerhalb des laufenden Linux ist der Dovecot-User ja ein unbekannter, nicht existierender User, denn dieser hat nur für Dovecot eine Bedeutung.

```
root@raspi3:/etc/dovecot
# touch /etc/dovecot/alias_maps
# chown root:dovecot /etc/dovecot/alias_maps
# chmod 644 /etc/dovecot/alias_maps
# ne /etc/dovecot/alias_maps
```

```
# Assigns virtual dovecot-user to a suitable linux-user
#=========================================================

thomas.addams_t     thomas
thomas.addams_g     thomas
tom.addams          thomas

silvia.addams_t     silvia
silvia.addams_g     silvia

manuel.addams_t     manuel
manuel.addams_g     manuel

familie             familie
```

Die nächste zu erstellende Datei ist zum jetzigen Zeitpunkt erst mal nur ein Dummy, die erst zu einem späteren Zeitpunkt endgültig erstellt wird. Es handelt sich hierbei um ein Script, welches später von Dovecot bei der Anmeldung eines User gestartet wird, um den tatsächlichen Transport der Emails vom ISP-Mailaccount durchzuführen. Da wir diese Funktion wegen der fehlenden Getmail-Konfiguration aber im Moment noch nicht nutzen können, soll Dovecot einfach mit der Default-Bearbeitung fortfahren und nicht auf "Fehler laufen", weil das Script zwar in der Conf angegeben ist, es aber noch nicht existiert. Dieser Dummy gewährleistet, dass der reguläre Login eines Users ohne Unterbrechung durchgeführt wird. Man könnte auch die entsprechenden Zeilen in der dovecot.conf einkommentieren, aber dann darf später man nicht vergessen, diese Comment-Tags wieder zu entfernen.

```
root@raspi3:/etc/dovecot
# echo -e '#!/bin/bash\nexec "$@"' >/usr/local/bin/getmail_eventhandler

# chmod +x /usr/local/bin/getmail_eventhandler
# chown root:root /usr/local/bin/getmail_eventhandler
```

Der erste Schritt zur richtigen Inbetriebnahme des eigentlichen Programms ist wie immer gleich, die obligatorische Sicherung der Original-Conf. Entweder wird sie mit dem MC kopiert oder klassisch von Hand:

```
root@raspi3:/etc/dovecot
# mv /etc/dovecot/dovecot.conf /etc/dovecot/dovecot.conf.orig
```

Nun erstellen wir unsere eigene Konfiguration für dovecot:

Ein paar Anmerkungen vorab zum Inhalt der Konfigurationseinstellungen

1. Die Anmeldung mit plaintext-PWD ist nicht erlaubt, siehe disable_plaintext_auth = yes

2. auth_mechanisms = plain login ist kein Widerspruch zu 1, da der Traffic als Ganzes via TLS verschlüsselt ist. Das Password hier noch einmal zu verschlüsseln bedeutet ein verschlüsseltes Password in bereits verschlüsselter Datenübertragung zu senden. Das ist natürlich Unsinn.

3. Postfix ist so konfiguriert, dass es eine Berechtigungsprüfung gegen die Dovecot-UserDB durchführt, siehe service auth{}. Wer keine Dovecot-Berechtigung hat, kann auch keine Mail senden.

4. Einige Parameter sind **Rot** markiert. Diese Parameter können oder müssen ggf. nach Bedarf angepasst werden. Die Bedeutung des Inhaltes dieser Parameter ist selbsterklärend.

5. Zur späteren Diagnose bei Problemen oder einer Fehlersuche kann man bei Bedarf die hier folgenden Parameter aktivieren, die anfangs per default inaktiv sind. Dadurch werden weitere und umfassendere Journaleinträge generiert, die möglicherweise Hinweise auf Fehler oder Probleme enthalten:

```
#auth_verbose=yes                       # enables logging all failed authentication attempts.

#auth_debug=yes                         # enables all authentication debug logging (also
                                        # enables auth_verbose). Passwords are logged as
                                        # <hidden>.

#auth_debug_passwords=yes               # does everything that auth_debug=yes does, but it
                                        # also removes password hiding (but only if you are
                                        # not #using PAM, since PAM errors aren't written
                                        # to Dovecot's own logs).

#mail_debug=yes                         # enables all kinds of mail related debug logging,
                                        # such as showing where Dovecot is looking for mails.

#verbose_ssl=yes                        # enables logging SSL errors and warnings. Even
                                        # without this setting if connection is closed
                                        # because of an SSL error, the error is logged as
                                        # the disconnection #reason.

#auth_verbose_passwords=no|plain|sha1   # If authentication fails, this setting logs the used
                                        # password. If you don't really need to know what the
                                        # password itself was, but are more interested in
                                        # knowing if the user is simply trying to use the
                                        # wrong password every single time or if it's a brute
                                        # force attack, you can set this to "sha1" and only
                                        # the SHA1 #of the password is logged. That's enough
                                        # to know if #the password is same or different
                                        # between login #attempts.
```

Es ist sicher möglich, die Conf's und die Programme hier im PDF zu markieren, zu kopieren und dann via Paste im Texteditor einzufügen. Bei Seitenwechseln sind aber Kopierfehler vorbestimmt, da auch die Seitenzahlen kopiert und eingefügt werden. Je nach PDF-Reader werden Einrückungen entfernt, was eine unübersichtliche Version ergibt. Ich empfehle deswegen, unbedingt alle relevanten Dateien aus dem Tar-File zu übernehmen und dort die Anpassungen vorzunehmen.

```
root@raspi3:/etc/dovecot
# touch /etc/dovecot/dovecot.conf
```

Sofern die neue Datei nicht ‚root‘ und 'dovecot' gehört oder andere Rechte gesetzt sind:
`Rechteeinstellung:  -rw- r-- r--  root:dovecot`

```
# chown root:dovecot /etc/dovecot/dovecot.conf
# chmod 644 /etc/dovecot/dovecot.conf
```

Zur Erinnerung: alle notwendigen Dateien können mit diesem Archiv herunter geladen werden:
http://www.thlu.de/Public/TomsMailserver.tar

Die Konfigurationseinsteillungen des Mailservers:

```
# ne /etc/dovecot/dovecot.conf
```

```
# Version: 4.3
# Date:    25.07.2022

auth_verbose_passwords=sha1
disable_plaintext_auth = yes
mail_privileged_group = vmail
hostname = mail
mail_location = maildir:/media/SSD/Mail/Imap:UTF-8
mail_home = /media/SSD/Mail/Sieve/%n

# Nur User vmail UID=5000/GID=5000 ist erlaubt
first_valid_uid = 5000
last_valid_uid = 5000

first_valid_gid = 5000
last_valid_gid = 5000

protocols = imap lmtp
#--------------------------------------------------------------------------------
# Bei Login eines Users in sein Dovecot-Postfach ein Script starten, dass
# für dieses Email-Konto via Getmail die Emails vom ISP-Mail-Konto abruft.

service imap {
  executable = imap imap-postlogin
}

service imap-postlogin {
  executable = script-login /usr/local/bin/getmail_eventhandler

# user = $default_internal_user
  user = root
  unix_listener imap-postlogin {
  }
}
#--------------------------------------------------------------------------------
# Zustellung lokaler EMails innerhalb des LAN abwickeln, ohne Beteiligung des
# Internets. Erstellt einen Unix-Socket zur Interprozesskommunikation zwischen
# Dovecot und Postfix. (siehe /etc/postfix/rules/check_local_transport_maps)

service lmtp {
 unix_listener /var/spool/postfix/private/dovecot-lmtp {
   group = postfix
   mode = 0600
   user = postfix
  }
}
```

```
#---------------------------------------------------------------------------------
# Dovecot User–Accountmanagement, Zugangsdaten sind in PWD-File hinterlegt

passdb {
  driver = passwd-file
  args = scheme=plain-md5 username_format=%n /etc/dovecot/passwd
}
userdb {
  driver = passwd-file
  args = username_format=%n /etc/dovecot/passwd
}
#---------------------------------------------------------------------------------
# Postfix smtp-auth. dovecot ist das authentication backend für Postfix
# Zum Senden von Emails über Postfix werden das starttls-Kommando und die
# Dovecot-Anmeldedaten verlangt. Unberechtigte Anwender können nicht senden.
# Die Interprozesskommunikation zwischen Postfix und Dovecot findet über
# diesen von Dovecot etablierten Socket statt.
# (siehe postfix.conf -> smtpd_sasl_type = dovecot)

service auth {
  unix_listener /var/spool/postfix/private/auth {
  mode = 0666
  }
}
auth_mechanisms = plain login

#---------------------------------------------------------------------------------
# Getmail führt bei der Abholung der Emails eines Users ein SetUID auf den User
# vmail durch und ermöglicht hiermit ein Linux-Schreibrecht auf die Mail-
# Verzeichnisse des Users.
# service auth erlaubt den Usern der Gruppe "mail" sich über den Socket
# auth-userdb zu authentifizieren und die Zustellung in das eigene Postfach
# via dovecot LDA durchzuführen.

service auth {
  unix_listener auth-userdb {
    group = vmail
    mode = 0660
  }
}
#---------------------------------------------------------------------------------
# Socket-Permissions für Gruppe vmail setzen. Troubleshooting für
# net_connect_unix(/var/run/dovecot/stats-writer) failed: Permission denied
#   ls /var/run/dovecot/stats*
#   srw-r----- 1 root vmail 0 2019-09-25 11:59 /var/run/dovecot/stats-reader
#   srw-rw---- 1 root vmail 0 2019-09-25 11:59 /var/run/dovecot/stats-writer

service stats {
    unix_listener stats-reader {
        group = vmail
        mode = 0640
    }

    unix_listener stats-writer {
        group = vmail
        mode = 0660
    }
}
#---------------------------------------------------------------------------------
# Prot. lda  = führt die Sieve-Scripte für Getmail->Dovecot-Delivered Mails aus
# Prot. lmtp = dto. für via Postfix lokal gesendete und empfangene Mails
# Plugin beschreibt die Orte, an denen die Sieve-Scripte gespeichert sind

protocol lda {
  mail_plugins = $mail_plugins sieve
```

```
}
protocol lmtp {
  mail_plugins = $mail_plugins sieve
}

plugin {
   sieve_dir = /media/SSD/Mail/Sieve/%n
   sieve = /media/SSD/Mail/Sieve/%n/.dovecot.sieve

   sieve_global_dir = /media/SSD/Mail/Sieve
   sieve_default = /media/SSD/Mail/Sieve/default.sieve
}
#--------------------------------------------------------------------------------
# Mailbox-Defaults - autocreated mailboxes are created lazily to disk only
# when accessed for the first time.

namespace inbox {
  inbox = yes

  mailbox Trash {
    auto = subscribe
    special_use = \Trash
  }
  mailbox Drafts {
    auto = subscribe
    special_use = \Drafts
  }
  mailbox Sent {
    auto = subscribe
    special_use = \Sent
  }
  mailbox Spam {
    auto = subscribe
    special_use = \Junk
  }
}
#--------------------------------------------------------------------------------
# Zertifikat und Key, verschlüsselt den Datenverkehr bzw. lässt
# unverschlüsselten Datenverkehr nicht zu.
# Mögliche Werte für die Protokoll-Festlegung sind: SSLv3,TLSv1, TLSv1.1, TLSv1.2

ssl = required
ssl_min_protocol=TLSv1.2
ssl_prefer_server_ciphers = yes
ssl_cipher_list =
EECDH+ECDSA+AESGCM:EECDH+aRSA+AESGCM:EECDH+ECDSA+SHA384:EECDH+ECDSA+SHA256:EECDH+aRS
A+SHA384:EECDH+aRSA+SHA256:EECDH+aRSA+RC4:EECDH:EDH+aRSA:!aNULL:!eNULL:!LOW:!3DES:!
MD5:!EXP:!PSK:!SRP:!DSS:!RC4
ssl_dh   = </etc/dovecot/certs/dh4096.pem
ssl_cert = </etc/dovecot/certs/server_ca_chain-cert.pem
ssl_key  = </etc/dovecot/certs/serverkey.pem

#================================================================================
# Ende
```

Die jetzt noch fehlenden letzten Handgriffe schließen das Kapitel Dovecot ab. Wir überprüfen, ob die Dateien bestehen, welchen Inhalt sie haben und ggf. aktualisieren wir den Inhalt. Die folgenden Dateien also entweder anlegen, oder korrigieren, wenn schon vorhanden, oder Werte eintragen:

```
root@raspi3:~
# ne /etc/hostname
raspi3
```

```
# ne /etc/hosts
127.0.0.1   localhost raspi3
127.0.1.1   mail
```

Wenn die folgenden Dateien schon vorhanden sind, muss ggf. auch der Inhalt geändert werden:

```
root@raspi3:/
# ne /etc/mailname
mail
```

Falls /etc/mailname nicht existiert, muss sie angelegt werden:

```
root@raspi3:/
# echo 'mail' >/etc/mailname
```

Wie schon zuvor erwähnt, möchte ich noch einmal darauf hinweisen: der Hostname muss nicht 'raspi3' lauten, so heißt hier nur mein Testrechner. '*raspi3*' ist also nur der Hostname meines Testsystems, auf dem und mit dem ich diese Dokumentation prüfe und ausführe. Wenn der Hostname ein anderer ist, z.B. *server*, oder *mailserver*, oder *lanserver*, oder *meinserver*... alles ok, der PC kann heißen wie er will, er sollte nur via DNS im lokalen Netz bekannt sein. Sofern hier eine Änderung durchgeführt wird, empfehle ich den Rechner einmal neu zu starten, so das auch im Netzwerk der korrekte Hostname bekannt ist.

11. Benutzer-Verwaltung – ein Überblick über die Zusammenhänge

Zur abschließenden Einrichtung des Mailservers fehlt jetzt noch die virtuelle Benutzerverwaltung. An dem Punkt ist allerdings vorher eine wichtige Entscheidung mit möglicherweise Auswirkungen auf zukünftige Änderungen zu treffen: Soll es überhaupt eine virtuelle Benutzerverwaltung sein oder soll für die Anmeldung auf das Emailkonto auch der sowieso vorhandene Linux-Account genutzt werden?

Ich habe mich nach reiflicher Überlegung entschieden, für die Anmeldung an das Mailsystem nicht den Linux-Account zu verwenden, sondern dafür eigene virtuelle Benutzer anzulegen. Ein mögliches Problem mit Anmeldedaten und in der Folge davon vielleicht ein beeinträchtigter Datenschutz und Serverkompromittierung entsteht meiner Einschätzung nach durch die mobile Hardware der Anwender. Leider entzieht es sich ein ganzes Stück weit einer übergreifenden Kontrolle, an welchem Ort die Anwender mit welcher Netzverbindung und wie umsichtig sie sich mit dem Server zuhause verbinden. Erfolgt die Anmeldung an den Mailserver via OpenVPN kann m.E. nicht viel passieren. Erfolgt aber ein Anmeldeversuch bevor der sichere Tunnel etabliert ist, vielleicht weil es gerade vergessen wurde, findet möglicherweise ein Anmeldeversuch über einen unsicheren (vielleicht auch unseriösen) Hotspot ohne verschlüsseltem Tunnel-Traffic statt.

Das Horrorszenario ist natürlich, das durch Fehler beim Handling des Anwenders, z.B. die Oma, möglicherweise gleichzeitig sowohl der DynDNS-Name meines Servers abgegriffen werden kann und vielleicht auch noch die Linux-Anmeldedaten beim Anmeldeversuch an den Mailserver. Diese Gefahr hat mich gerade auch vor dem Hintergrund der Sambashares und dem Zugriff aller Familienmitglieder auf wichtige „gemeinsame Daten“ einigermaßen beunruhigt.

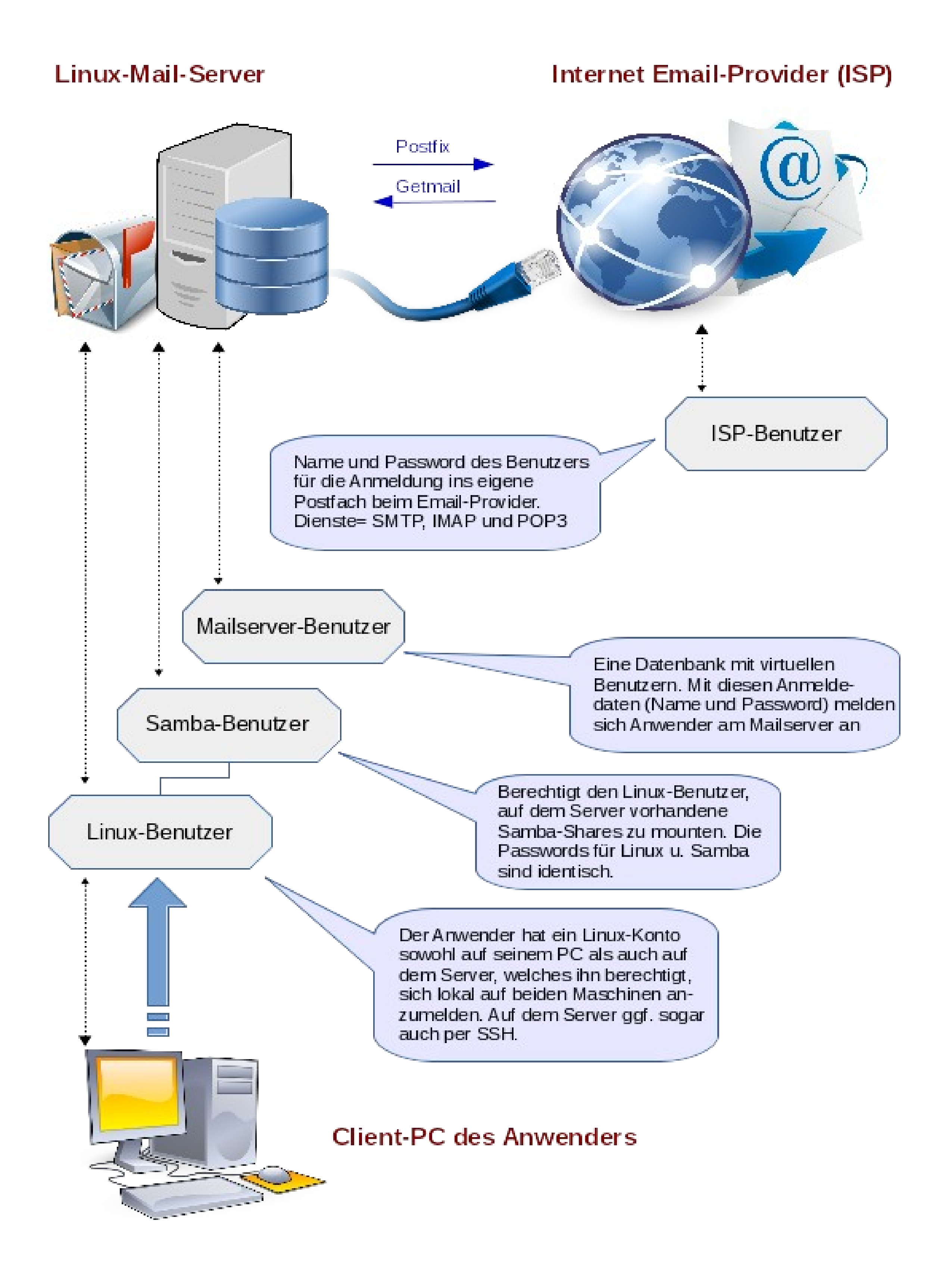
Bei diesen Gedanken schien es mir der bessere Weg zu sein, für die Anmeldung an den Mailserver nicht die Linux-Anmeldedaten zu verwenden, die ja draußen und unterwegs sowieso nicht einfach so verwendet werden können, sondern für jeden User einen virtuellen User mit Anmeldedaten nur zum Mailserver anzulegen. Darüber hinaus ist die Benutzerverwaltung aufgrund des Umstands „*Familie*“ im Wesentlichen auch statisch, was bedeutet, einen ständigen Wechsel oder „Personal-Fluktuation“ gibt es hier nicht. Der „virtuelle User“ ist hier also kein Nachteil. Bei weiteren Überlegungen war das dann einer der Hauptgründe, warum ich nun auch die Möglichkeit verworfen habe, das Familienmitglieder über einen Password-Dialog flexibel und nach eigenem Ermessen ihre Zugangspasswörter ändern können. Das verhindert wiederum die Notwendigkeit weitere Software installieren zu müssen, eben für den Password-Dialog einen Web-Server und das zusätzlich noch notwendige Web-Interface für das Postfach und den Mailserver, welches dann über den Browser bedient werden kann. Ein wichtiger Aspekt war also, der Verzicht auf diese Funktionen reduziert gravierend auch das Maß an Sicherheitslücken durch meine Customizing-Fehler oder durch latent vorhandene Exploits durch eine sowieso potentiell unsichere Software, die tatsächlich enorm viel Sachkenntnis für einen sicheren Betrieb erfordert.

Darüber hinaus ist weder der Kenntnisstand noch das Risiko- und Schadensbewusstsein verschiedener User so ausgebildet, dass sie sich über alle technischen Belange im Klaren sind und überhaupt einen solchen Wunsch zur individuellen PWD-Änderung äußern. Das Motto ist überwiegend *„ich will, dass es ohne Probleme, ohne das es mich nervt und ohne kompliziert zu sein einfach so funktioniert“.* Wir sind eben kein Unternehmen mit Revision und Datenschutzbeauftragten, sondern einfach nur *Familie*… also muss ich abwägen, wie ich die Interessen (meine und die der Anwender) unter einen Hut bringe. Man kann wohl alles machen, man muss es aber nicht…..
Im Endeffekt ist es bei uns nun so, das einmalig eingerichtete lokale Accounts eine eher lange Lebensdauer haben, was meinen Wartungsaufwand natürlich gering hält. Mein Fazit nach längerer Zeit der Beobachtung ist, es ist eine pragmatische Lösung, die Aufwand und Resultat in einem gut ausgewogenen Verhältnis hält.

Jetzt ist auch der richtige Zeitpunkt, um mal über eine tatsächliche Schwäche dieses Konzeptes zu sprechen, und zwar der Datenschutz und das Briefgeheimnis. Ja, es stimmt, mir sind sämtliche Zugangs- und Anmeldedaten aller Anwender bekannt bzw. sie sind auf einem Datenblatt notiert. Das wird später bei der Einrichtung von Postfix ein weiteres Mal deutlich erkennbar. Aber das war zum Teil auch schon in der Vergangenheit unvermeidbar und passierte sogar zwangsläufig durch die Aufforderung *„Mach mir das mal so fertig, dass es funktioniert.“* Dieser Umstand ist hier allgemein bekannt und es interessiert keinen. Dabei darf man nicht vergessen, dass ich als „Herr und Admin“ der Hardware sowieso Zugang zu allen Festplattenbereichen habe, allein schon wegen der notwendigen Backups. Tja, so ist es halt. Glücklicherweise habe ich weder eine krankhafte Neugier noch einen

therapie-würdigen Kontrollzwang – ich schreib nur gerne Anleitungen :-). Stattdessen bin ich mit einer grenzenlosen Gleichgültigkeit gesegnet, soweit es die IT-technischen Lebensbanalitäten der anderen angeht. Nun könnte man bemerken, ich hätte sogar die Position einer familiären NSA… ja richtig… aber es interessiert mich einfach nicht.

Um also die Zusammenhänge des Benutzermanagements einmal komplett zu verstehen, schauen wir uns die folgende Übersicht an. Jede Benutzerebene enthält ein eigenes Password, welches sowohl identisch zu einem anderen sein kann, als auch unterschiedlich zu allen anderen. Für das Linux-Konto und die Samba-Shares verwende ich bei den Anwendern z.B. jeweils das gleiche Password.



Da es sich in unserem *Familienbetrieb* ja um eine sehr übersichtliche Anzahl von Benutzern handelt, ist der einfachste Weg, die verschlüsselten Passwörter einfach in einer simplen Textdatei zu hinterlegen. Die Alternative wäre, dafür auch noch wieder eine reguläre Datenbank mit Backend und dem natürlich dann notwendigen Frontend einzurichten. Aber das steht hinsichtlich des Aufwandes in keinerlei sinnvollen Verhältnis zum Resultat. Darüber hinaus sind es nur wieder weitere zu wartende Programme, mit vielleicht auch weiteren schlummernden Sicherheitslücken und Risiken. Und weil die lokalen Zugangsdaten hier sowieso weitestgehend statisch und langlebig sind, reicht die Hinterlegung in einer strukturiertem Textform für uns völlig aus.

Dovecot bringt mit „*doveadm*" ein eigenes Werkzeug zur Verschlüsselung von Passwörtern mit. Ich habe darauf und auf der Entscheidung zum Textformat basierend für uns folgende Anmelde- und Password-Regeln festgelegt:

- alle lokalen Postfächer eines Users auf unserem Mailserver haben das gleiche Password
- der Anmeldename zum jeweiligen Postfach des Mailserver ist die lokale Email-Adresse
- das Mailserverpassword besteht aus dem Linux-Password des Users und einem Mailserverprefix
  Als Prefix habe ich *Mailsys-rpi3.* festgelegt.

Nehmen wir also an, ich könnte mich mit dem Anmeldenamen „thomas" und dem Password „toms_passwd" im Linux des Servers anmelden, dann ist „toms_passwd" gleichzeitig auch das Password für die Sambashares. Man kann unterschiedliche verwenden, meiner Meinung nach ist das aber nicht sinnvoll und bringt keinen wirklichen Gewinn an Sicherheit.

Nehmen wir weiter an, TomL.de wäre meine eigene Internet-Domain, thomas.addams@toml.de und thomas.addams@gmx.de wären meine mit vollständigen Namen personifizierten und damit seriösen Emailadressen, die via POP3 und Getmail in den lokalen Mailserver auf die lokalen Emailadressen thomas.addams_t@mail und thomas.addams_g@mail übertragen werden, so erfolgt die lokale Anmeldung im Mailserver für die beiden lokalen Postfächer/Usernamen *beide Mal mit dem* gleichen Password *„mailsys-rpi3_ toms_passwd"*

Der Namens-Erweiterung im lokalen Postfach mit _t und _g ist notwendig, weil es keine 2 Postfächer am gleichen Speicherort mit gleichem Namen geben kann. Weil also der Postfachname in beiden Fällen thomas.addams lautet, wird zur Unterscheidung **_t** für die Domain TomL und **_g** für GMX angefügt. Damit ist die notwendige Eindeutigkeit hergestellt.

Hier ist eine leichter zu überblickende Übersicht, an denen man die Regeln erkennen kann, wenn jeder User mindestens 2 mit seinem vollen Namen personifizierte „seriöse" Postfach-Accounts im Internet hat. Natürlich werden die getrennten Internet-Accounts auch hier in unserem lokalen Webserver als getrennte Postfächer behandelt.

Ziel	**Anmeldename/Adresse**	**Password**
Linux-Anmeldung u. Samba-Shares	thomas	toms_passwd
Provider-Mail-Account bei toml.de	thomas.addams@toml.de	irgendeingeheimespwd
Lokales Postfach	thomas.addams_t@mail	Mailsys-rpi3.toms_passwd
Provider-Mail-Account bei gmx.de	thomas.addams@gmx.de	irgendeinanderesgeheimespwd
Lokales Postfach	thomas.addams_g@mail	Mailsys-rpi3.toms_passwd

Linux-Anmeldung u. Samba-Shares	silvia	123abc
Provider-Mail-Account bei toml.de	silvia.addams@toml.de	eineigenespwd
Lokales Postfach	silvia.addams_t@mail	Mailsys-rpi3.123abc
Provider-Mail-Account bei gmx.de	silvia.addams@gmx.de	einandereseigenesgeheimespwd
Lokales Postfach	silvia.addams_g@mail	Mailsys-rpi3.123abc

Usw., für alle anderen Anwender nach den gleichen Regeln!

Wenn sich die lokalen Postfachnamen grundsätzlich unterscheiden, ist die oben verwendete Erweiterung mit _t und _g natürlich unnötig. Das ist wirklich nur dann notwendig, wenn zwei oder mehr Postfachnamen auf Grund der Umstände tatsächlich identisch sind.

Da ich hier ständig von Namen spreche und auf deren Bedeutung hinsichtlich Eindeutigkeit hinweise, bitte ich auch noch einmal daran zu erinnern, dass „raspi3“ in dieser Anleitung nur der Hostname meines Test-Mailservers war – siehe Software-Installation (Kap. 7) und dovecot.conf (Kap. 9).

Warum das alles auf diese Weise? Ganz einfach, ich will mir nicht ständig neue Passwörter ausdenken müssen. Ebenso wenig will ich kryptische maschinelle Passwörter verwenden. Gerade auch dann, wenn man Passwörter am Smartphone eingeben möchte, oder am Laptop, wo man nicht mal eben schnell was markieren, kopieren und einfügen kann, ist es einfach, wenn man auf gewisse Regeln zurückgreifen kann. Insbesondere auch dann, wenn man das weiter oben erwähnte Datenblatt mal nicht gerade bei der Hand hat. Nach diesen Regeln habe ich wenigstens eine kleine Chance, bei Wartungsarbeiten durch einfache Ableitung beim Blick auf meine Anmeldung die Anmeldedaten eines anderen Anwenders zu ermitteln.

Es gibt 2 Wege, von Dovecot ein Password generieren zu lassen, einmal im Dialog eines Password-Chats und alternativ durch einen Befehl. Ich betrachte hier nur die schnellere Befehlsvariante.

Entweder einzeln User für User, hier mit meinem Password aus dem obigen Beispiel:

```
doveadm pw -s plain-md5 -p 'Mailsys-rpi3.toms_passwd'
```

Oder als 1 Befehl gleich für mehrere Benutzer in einer Rutsche abgesendet:

```
echo thomas;   doveadm pw -s plain-md5 -p 'Mailsys-rpi3.toms_passwd';\
echo silvia;   doveadm pw -s plain-md5 -p 'Mailsys-rpi3.123abc';\
echo manuel;   doveadm pw -s plain-md5 -p 'Mailsys-rpi3.mnemonic';\
echo familie;  doveadm pw -s plain-md5 -p 'Mailsys-rpi3.familypwd'
```

Mit folgender Ausgabe im Terminal:

```
thomas
{PLAIN-MD5}6e734610019959efc1bb814b6bf9056d
silvia
{PLAIN-MD5}83e7d701351c2e87b15eb1e4c8cf9206
manuel
{PLAIN-MD5}c3285509693b11d888a1e0e18007ce82
familie
{PLAIN-MD5}defb0fcc1b3207938abab35f89601432
```

Der korrekte Result-Hash kann mit dem folgenden Befehl verifiziert werden:

```
doveadm pw -t '{PLAIN-MD5}6e734610019959efc1bb814b6bf9056d' -p 'Mailsys-rpi3.toms_passwd'
{PLAIN-MD5}6e734610019959efc1bb814b6bf9056d (verified)
```

Die Passwörter werden nun in die Textfile-Password-DB von Dovecot übertragen. Dazu zuerst die unten beschriebene Datei erstellen, die Rechte setzen und dann im Editor die PWD‘s einzeln per Copy/Paste einfügen und die Datensätze pro Zeile vervollständigen.

Es ist auch möglich, einfach die Beispiel-Datei aus dem Tar-File zu übernehmen und Zeile für Zeile anzupassen, fehlende Zeilen hinzuzufügen und überschüssige zu löschen. Anschließend, wenn die Datei fertig erstellt ist, müssen aber auf jeden Fall die Rechte korrigiert werden.

```
root@raspi3:/etc/dovecot
# touch /etc/dovecot/passwd
# chown root:dovecot /etc/dovecot/passwd; chmod 640 /etc/dovecot/passwd
```

```
Rechteeinstellung:  -rw- r-- --- root:dovecot   passwd
```

Bedauerlicherweise ist es nicht möglich, das Beispiel auf dieser Seite in gesamter Breite in normaler Schriftgröße darzustellen. Aus diesem Grund und um die Übersichtlichkeit beizubehalten musste ich das Password leider einkürzen. Aber das hat natürlich keine Auswirkung auf den tatsächlichen Inhalt. Und selbstverständlich muss das Password in der endgültigen Version vollständig übernommen werden.

Die Zeilen in dieser Textfile-Password-DB enthalten die Einträge für jeweils ein Postfach eines Mailusers, die zusammenhängend als eine Zeile gespeichert werden muss. Sinnvollerweise auch ohne Leerzeichen. Das hier erstellte Muster enthält die Leerzeichen nur scheinbar der Übersichtlichkeit wegen. In Wirklichkeit sind diese Leerzeichen das u.g. Feld Comment User-Info.

```
# ne /etc/dovecot/passwd
```

```
thomas.addams_t:{PLAIN-MD5}007fa0df:vmail:vmail:  :::userdb_mail=maildir:/media/SSD/Mail/Imap/thomas/thomas.addams_t
thomas.addams_g:{PLAIN-MD5}6e734610:vmail:vmail:  :::userdb_mail=maildir:/media/SSD/Mail/Imap/thomas/thomas.addams_g

silvia.addams_t:{PLAIN-MD5}83e7d701:vmail:vmail:  :::userdb_mail=maildir:/media/SSD/Mail/Imap/silvia/silvia.addams_t
silvia.addams_g:{PLAIN-MD5}83e7d701:vmail:vmail:  :::userdb_mail=maildir:/media/SSD/Mail/Imap/silvia/silvia.addams_g

manuel.addams_t:{PLAIN-MD5}c3285509:vmail:vmail:  :::userdb_mail=maildir:/media/SSD/Mail/Imap/manuel/manuel.addams_t
manuel.addams_g:{PLAIN-MD5}c3285509:vmail:vmail:  :::userdb_mail=maildir:/media/SSD/Mail/Imap/manuel/manuel.addams_g

familie:{PLAIN-MD5}defb04325sdffcc1:vmail:vmail:  ::::userdb_mail=maildir:/media/SSD/Mail/Imap/familie
```

Noch ein Hinweis zur Erinnerung: Postfix authentifiziert den User vor dem Senden seiner Email über einen Socket ebenfalls mit dieser User-DB. Das bedeutet: kein Eintrag hier = kein gültiges Password, kein Email senden…. siehe dovecot.conf

Die Dateistruktur – Beschreibung der Datenfelder:
username:password : uid:gid : Comment User-Info :home:shell : extra_fields

Feldname	Bedeutung	Anwender- und Postfach-Bezogener Inhalt
username:password	Linux-IDs des Users	thomas.addams_t:{PLAIN-MD5}007f1a0dfacb7869688c011d1f02288b:
uid:gid	Überschreibt Global-Mail-Settings für UID/GID	vmail:vmail:
Comment User-Info	ungenutzt	:
home	ungenutzt	:
shell	ungenutzt	:
extra_fields	enthält den Pfad zum Postfach	userdb_mail=maildir:/media/SSD/Mail/Imap/thomas/thomas.addams_t

Sofern keine Fehler in der bisherigen Einrichtung gemacht wurden, müsste es jetzt möglich sein, sich nach einem Neustart des Servers von Thunderbird aus zu allen in der Password-Datei definierten Postfächern anzumelden. Nach den schon umfangreichen Linux- und User-spezifischen Einstellungen ist hiermit nun das zweite Kapitel erfolgreich beendet, die Dovecot-Installation ist hier abgeschlossen. Aber ich empfehle, auf diesen Test zu verzichten… denn viel ist nicht zu sehen, Emails senden und empfangen ist auch noch nicht möglich. Es fehlen ja noch die Postfix-, Getmail- und LDA-Installation.

Wir können nun aber auch auf die Schnelle einen ganz einfachen Test mit Openssl von einem Client-PC mit installiertem Linux-Betriebssystem durchführen, um festzustellen, ob überhaupt eine fehlerfreie Verbindung zum Dovecot-IMAP-Mailserver möglich ist und welche Antworten er liefert. Ein solcher Test wird vermutlich auch mit einem Windows-System als Client-PC möglich sein, aber damit möchte ich mich eigentlich nicht weiter befassen, deswegen hier nur ein Beispiel unter Debian:

Die folgenden Befehle sind nacheinander und einzeln abzusetzen, der erste Befehl findet noch in der Linux-Shell statt, die folgenden Befehle interaktiv in einer Openssl-Shell:

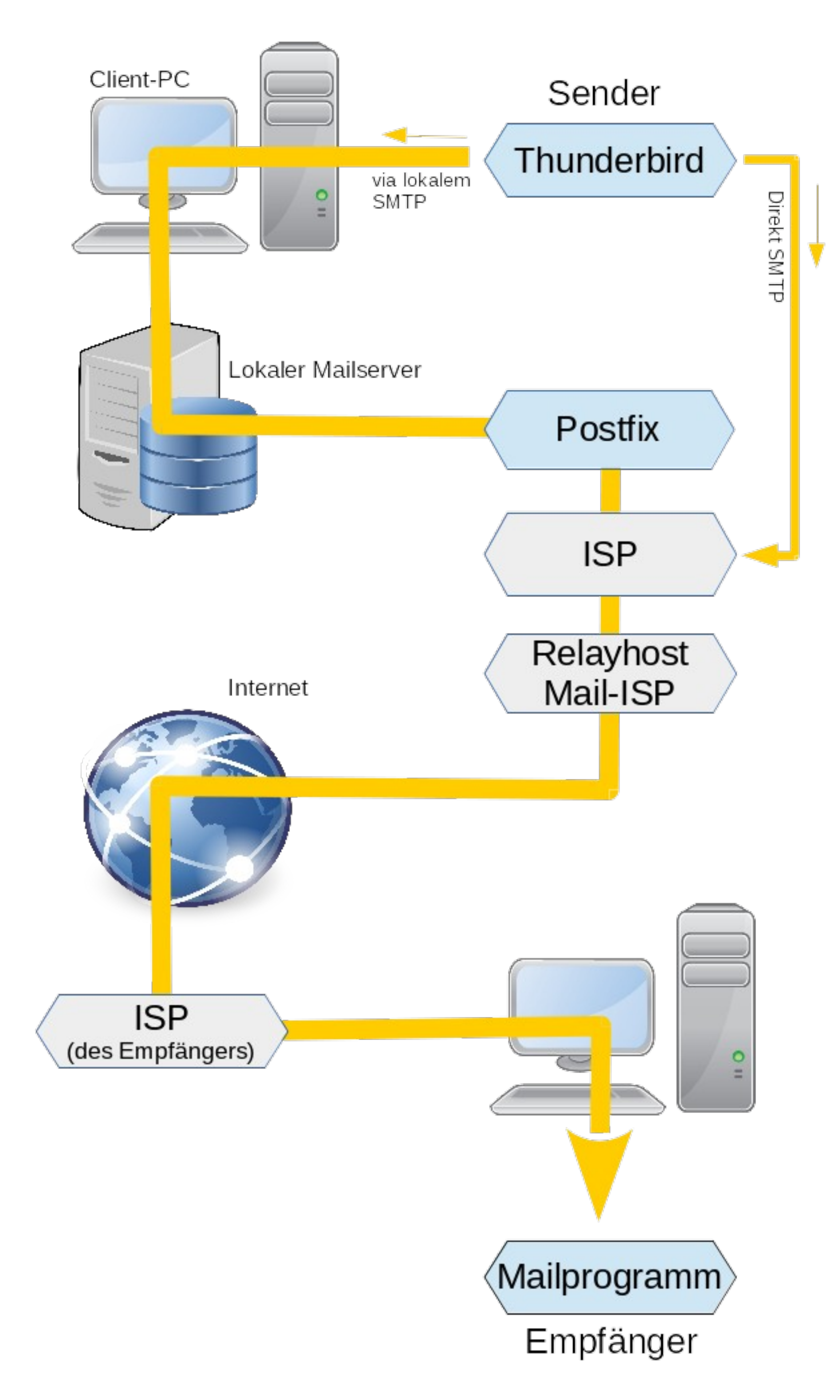
```
$ openssl s_client -connect mail:993 -tls1_3
tag login thomas.addams_t@mail Mailsys-rpi3.toms_passwd
tag list "" "*"
tag logout
* BYE Logging out
tag OK Logout completed (0.001 + 0.000 secs).
Closed
$
```

Das Kommando „tag list“ wird die im Postfach des angemeldeten Users jetzt vorhandenen/bestehenden Postfach-Ordner auflisten. Das kann zum jetzigen Zeitpunkt allerdings noch sehr spartanisch aussehen, eben weil das Server-Postfach noch gar keinen Kontakt zu einem echten Mail-Client-Programm wie z.B. Thunderbird hatte. Im Regelfall werden bei einem Erstkontakt automatisch notwendige Ordner eingerichtet. Wenn hier jedoch kein Fehler berichtet wird, würde ich davon ausgehen, dass alles in Ordnung ist.

12. Postfix – Einrichtung des Mail-Transport-Agents

Postfix ist das Programm, welches unsere mit Thunderbird erstellten Emails auf die Reise durchs Internet bringt. Dabei besteht die bemerkenswerte Besonderheit, dass von unserem lokalen Mailserver Emails eigentlich gar nicht so einfach versendet werden können, zumindest nicht direkt von uns übers Internet an einen Empfänger irgendwo auf der Welt.

Einer der wesentlichen Gründe ist, dass die Netzwerk-IP-Adresse unseres Servers eben nur eine lokale Adresse ist, vergeben von der DHCP-Funktion unseres Routers. Und weiterhin, das Privatkunden üblicherweise keine wirklich statische IP-Adresse bekommen, mit der die eigene private Domain dauerhaft eindeutig im Internet identifiziert werden kann.



Und sofern es sich sogar noch bei der internen IP-Adresse unseres Netzwerkes um eine IPv4-Adresse handelt, ist mit großer Sicherheit davon auszugehen, dass es weltweit vermutlich viele Endgeräte mit genau dieser gleichen Adresse gibt.

Eine eindeutige Identifizierung des Absenders und des Empfängers sind bei diesen Umständen also gar nicht möglich. Ohne das jetzt aber weiter zu vertiefen, akzeptieren wir einfach, dass der direkte Email-Versand durch unseren Server nicht möglich ist. Wir benötigen zum Senden eine Relais-Station, oder im Fachjargon einen Relayhost bzw. Smarthost. Für die Lösung dieses Problems verwenden wir zum Senden unserer Emails als Smarthost also einfach den SMTP-Server unseres Email-Providers.

Der Weg ist dann folgendermaßen: unser Client-PC mit Thunderbird sendet eine Email. Die Mail wird über den Port 25 an Postfix weitergeleitet und geht dann über die beiden ISP-Provider auf die Reise durchs Internet, bis sie dann irgendwann vom Empfänger abgerufen wird, auf seinem PC ankommt und gelesen werden kann.

Dabei verwenden wir nicht immer den gleichen Smarthost, also einen für alle, sondern wir verwenden für jeden einzelnen Benutzer den zu seiner Absenderadresse passenden Smarthost. Das sichert zuverlässig, dass gesendete Emails auf unserer Seite vor dem Internet solange wie möglich an den Absender gebunden sind und nicht den Smarthost oder gar den ISP-Sent-Ordner eines anderen Users berühren. Da die Technik und die systeminternen Abläufe insgesamt kaum zu durchschauen sind, habe ich mich für diese Eindeutigkeit entschieden… eine Eindeutigkeit, die ich selber durchs Customizing herstellen kann. Das bedeutet zwar etwas mehr Aufwand bei der Einrichtung, aber das war es mir wert.

Auch hier beginnen wir -wie zuvor bei der Einrichtung Dovecot- mit der obligatorischen Sicherung der Original-Conf

```
root@raspi3:/etc/postfix
# mv /etc/postfix/main.cf /etc/postfix/main.cf.orig
```

und erstellen uns eine eigene neue Datei, sowie ein neues Verzeichnis als Speicherort für weitere notwendige Dateien, die ich zuvor schon als Regelwerk bezeichnet habe. Bitte das ganze folgende Paket komplett markieren und als 1 Befehl absenden

```
mkdir -p /etc/postfix/rules;\
touch /etc/postfix/rules/refresh-rules;\
touch /etc/postfix/rules/check_generic_replace_from;\
touch /etc/postfix/rules/check_local_alias_maps;\
touch /etc/postfix/rules/check_local_transport_maps;\
touch /etc/postfix/rules/check_sender_filter;\
touch /etc/postfix/rules/check_sender_relayaccess;\
touch /etc/postfix/rules/get_sender_relayhost;\
touch /etc/postfix/rules/get_relayhost_passwd;\
touch /etc/postfix/rules/remove_header_sensitives;\
touch /etc/postfix/main.cf
```

Auf die angelegten Dateien für das Regelwerk hat außer „root“ niemand Lese- oder Schreibrechte. Um Missbrauch und Ausspähen zu unterbinden, werden deswegen sofort die Rechte korrekt gesetzt. Auch diese drei Zeilen zusammen markieren, kopieren und als 1 Befehl absenden:

```
chmod 700 /etc/postfix/rules;\
chmod 600 /etc/postfix/rules/*;\
chmod 700 /etc/postfix/rules/refresh-rules
```

Auf den folgenden Seiten werden wir ohne viel Umschweife die benötigen Informationen in die jetzt gerade erstellten Dateien einfügen.

Als erstes wird das kleine Script refresh-rules erstellt, welches es uns beim „Basteln“ oder späteren Änderungen ein wenig bequemer machen soll, die Originaldateien zu verschlüsseln und sie anschließend an die richtige Stelle zu kopieren. Man braucht das nicht zwingend, aber es vereinfacht die notwendigen Schritte. Ein für mich wichtiger Vorteil ist der Nebeneffekt, dass die angelegten Dateien einen einheitlichen Timestamp „Datei-Änderungszeit“ bekommen, damit ist auf den ersten Blick erkennbar, dass alles aktuell ist. Die passenden Prüfungs-Inhalte erfolgen als nächstes.

```
root@raspi3:/etc/postfix
# ne /etc/postfix/rules/refresh-rules
```

```
#!/bin/bash
#
# Support-Script to recreate rules
# Date   : 09.10.2019
# Version: 2.2
#=====================================================================

if [[ $EUID -ne 0 ]]; then
   echo -e "\nThis script must be run as root"
   exit 1
fi

cd /etc/postfix/rules
[ -f check_generic_replace_from ]   && postmap check_generic_replace_from
[ -f check_local_alias_maps ]       && postmap check_local_alias_maps
[ -f check_local_transport_maps ]   && postmap check_local_transport_maps
[ -f check_sender_filter ]          && postmap check_sender_filter
[ -f check_sender_relayaccess ]     && postmap check_sender_relayaccess
[ -f get_sender_relayhost ]         && postmap get_sender_relayhost
[ -f get_relayhost_passwd ]         && postmap get_relayhost_passwd

chown root:postfix /etc/postfix/rules/*.db
chmod 640 /etc/postfix/rules/*.db

[ -f check_generic_replace_from.db ] && mv -f check_generic_replace_from.db /etc/postfix
[ -f check_local_alias_maps.db ]     && mv -f check_local_alias_maps.db     /etc/postfix
[ -f check_local_transport_maps.db ] && mv -f check_local_transport_maps.db /etc/postfix
[ -f check_sender_filter.db ]        && mv -f check_sender_filter.db        /etc/postfix
[ -f check_sender_relayaccess.db ]   && mv -f check_sender_relayaccess.db   /etc/postfix
[ -f get_sender_relayhost.db ]       && mv -f get_sender_relayhost.db       /etc/postfix
[ -f get_relayhost_passwd.db ]       && mv -f get_relayhost_passwd.db       /etc/postfix

cp remove_header_sensitives /etc/postfix
chown root:postfix /etc/postfix/remove_header_sensitives
chmod 644 /etc/postfix/remove_header_sensitives

cd /etc
newaliases

echo -e "\nRestart Mailsystem! Continue?   (y/n): "
read CONFIRM && [ "$CONFIRM" = "n" ] && exit 0

clear
echo -e "Stop Mailsystem:\n"
[ -f /usr/sbin/dovecot ] && systemctl stop dovecot && systemctl status dovecot
[ -f /usr/sbin/postfix ] && systemctl stop postfix && systemctl status postfix

echo -e "\nMailsystem gestoppt!\nEinen kleinen Moment bitte!\n"
sleep 3

clear
echo -e "Start Mailsystem:\n"
[ -f /usr/sbin/dovecot ] && systemctl start dovecot && systemctl status dovecot
[ -f /usr/sbin/postfix ] && systemctl start postfix && systemctl status postfix
echo -e "\nMailsystem started!\n"

echo -e "\nCheck running Postfix-Configuration!"
postfix check
echo -e "Ready!\n"

exit 0
#=====================================================================
# Ende
```

check_generic_replace_from ist eine wichtige Replacement-Regel, die beim Sendevorgang den Mailheader untersucht und im Header unsere lokale Absender-Adresse gegen die internetkonforme Absender-Adresse (FQDN) des jeweiligen ISP-Postfaches austauscht.

```
root@raspi3:/etc/postfix
# ne /etc/postfix/rules/check_generic_replace_from
```

```
# Per-sender replacement of local "from" to Provider-"from"
#===========================================================

root@mail                thomas.addams@toml.de

thomas.addams_t@mail     thomas.addams@toml.de
thomas.addams_g@mail     thomas.addams@gmx.de
tom.addams@mail          tom.addams@gmail.com

silvia.addams_t@mail     silvia.addams@toml.de
silvia.addams_g@mail     silvia.addams@gmx.de

manuel.addams_t@mail     manuel.addams@toml.de
manuel.addams_g@mail     manuel.addams@gmx.de

familie@mail             thomas+silvia.addams@_toml.de
```

check_local_transport_maps verhindert, dass eine von einem lokalen Absender gesendete Email an einen lokalen Empfänger den Weg übers das Internet und den Mail-ISP geht. Die Email wird statt zum SMTP des Mail-ISP netzwerkintern an den LMTPD unseres dovecot-Mailservers transportiert und direkt in das Postfach des Empfängers zugestellt.

```
root@raspi3:/etc/postfix
# ne /etc/postfix/rules/check_local_transport_maps
```

```
# Lokal sender, local recipient - prevents transmission
# over the Internet, processes only local Traffic
#========================================================

mail  lmtp:unix:private/dovecot-lmtp
```

check_local_alias_maps übersetzt eine lokale Empfängeradresse, die mit dem Hintergrund einfacherer Bedienbarkeit in gekürzter Form eingegeben werden kann, in eine reguläre Empfänger-Email-Adresse unserer Mail-Domain. Darüber hinaus werden virtuelle Empfänger-Adressen des Linux-Systems, wie z.B. *root* oder *postmaster* ebenfalls übersetzt, um darüber System-Emails mit einer Systembenachrichtigung an das Postfach des Administrators zuzustellen - in diesem Beispiel ist das natürlich mein Postfach.

```
root@raspi3:/etc/postfix
# ne /etc/postfix/rules/check_local_alias_maps
```

```
# Translates shortened local recipients to regular local recipients
#===================================================================

root            thomas.addams_t@mail
root@mail       thomas.addams_t@mail
postmaster      thomas.addams_t@mail
mailer-daemon   thomas.addams_t@mail

thomas@mail     thomas.addams_t@mail
silvia@mail     silvia.addams_t@mail
manuel@mail     manuel.addams_t@mail
manu@mail       manuel.addams_t@mail
```

Eine besondere Möglichkeit für weitergehende Prüfungen auf vorhandene oder fehlende Berechtigung oder Plausibilität ist mit den smtpd-Restrictions gegeben. Um solche zusätzliche Überprüfungen bei einer zu sendenden Email durchzuführen, können nach eigenem Ermessen eine oder mehrere der folgenden Regeln bei der Verwendung von Postfix als Mailserver eingerichtet werden:

Name der Restriction	Zeitpunkt (resp. auslösendes Ereignis) für die Prüfung
`smtpd_client_restrictions`	`Prüfung nach dem Connect`
`smtpd_helo_restrictions`	`Prüfung nach dem HELO/EHLO`
`smtpd_sender_restrictions`	`Prüfung nach MAIL FROM:…`
`smtpd_relay_restrictions`	`Prüfung vor RCPT TO:…`
`smtpd_recipient_restrictions`	`Prüfung nach RCPT TO:…`
`smtpd_data_restrictions`	`Prüfung nach DATA`
`smtpd_end_of_data_restrictions`	`Prüfung nach erfolgter e-Mail Übertragung`
`smtpd_etrn_restrictions`	`Client leert via ETRN-Kommando die Mail-Queue`

Man muss hier respektvoll anerkennen, dass das durch die Postfix-Programmierer ohne jeden Zweifel klasse gelöst ist, aber leider ist auch anzumerken, dass es wirklich nicht ganz so leicht zu verstehen ist. Richtig kompliziert wird es durch die subjektive Perspektive auf eine Mail. Ich selber kann nämlich für *mein* Postfix sowohl Absender (sender) einer Mail sein, als auch Empfänger (recipient), je nachdem ob Postfix meine Email versendet oder eine ankommende Email an mein Postfach zustellt. Der Postfix-Server ist der initiale Host, wenn ich der Sender bin, aber es ist der finale Host, wenn ich Empfänger bin. Initial bedeutet, hier ist der Startpunkt, hier beginnt die Reise einer Email durchs Internet. Final bedeutet, hier ist der Endpunkt, das Ziel der Email-Reise durchs Internet, die Mail ist angekommen, es findet danach kein weiterer Transport statt.

Das ist dann noch leicht zu verstehen, wenn ich eine Mail ins Internet sende oder wenn ich eine Email aus dem Internet bekomme. In beiden Fällen ist mein Postfix-Server nur einmal beteiligt - einmal initial beim Senden, einmal final beim Empfangen. Vollkommen verwirrend wird es aber, wenn ich eine interne Email sende. In diesem Fall ist unser Postfix-Server zuerst der initiale Host und sofort auch der finale Host. Und wenn man oben auf die Restrictions schaut, sieht man, dass Prüfungen im übertragenen Sinne quasi nach oder bei Eintreten eines Ereignisses stattfinden, durch *"Prüfung vor"* und *"Prüfung nach"*. Und bei welchen Mail-Vorgängen greift dann überhaupt welche Prüfung? All das löst viele offene Fragen aus ...

... aber wir können trotzdem entspannt bleiben, es ist gar nicht so kompliziert, wie es scheint. Denn die für uns zunächst wichtigste Feststellung lautet: *"Man kann die Restrictions nutzen, aber man muss sie nicht nutzen!"* Postfix ist so ausgestaltet, dass es auch ohne Restrictions (Beschränkungen, i.ü.S. zusätzliche Regeln) bestens funktioniert. Die Restrictions sind eine Möglichkeit für zusätzliche Überprüfungen einer Email, die nur bei Eintreten eines bestimmten Zeitpunktes/Zustandes ausgeführt werden, ansonsten jedoch nicht. Beispielsweise wendet Postfix eine vorhandene Regel zur Prüfung des Absenders nicht an, wenn Postfix beim Empfang einer Mail der finale Host ist, selbst dann nicht, wenn Postfix die Mail bei der Zustellung ins Postfach des Empfängers selber noch einmal transportiert. Die Regel wird jedoch zuverlässig angewandt und berücksichtigt, wenn Postfix beim Senden einer Mail der initiale Host ist.

Wer sich jedoch insgesamt gegen eine Einrichtung dieser Beschränkungen entscheidet, lässt die hier von mir in diesem Konzept verwendeten 3 speziellen Restrictions einfach aus und kommentiert die entsprechenden Zeilen in der main.cf mit # - Postfix wird zweifellos trotzdem einwandfrei funktionieren.

Um welche Regeln handelt es sich, warum nur diese und warum überhaupt?

smtpd_sender_restrictions	Legt im Sinne einer Whitelist *(Blacklist geht auch)* fest, welche lokalen (!) Anwender den Postfix-Mailserver überhaupt verwenden dürfen.
smtpd_relay_restrictions	Legt im Sinne einer Whitelist *(Blacklist geht auch)* fest, wer einen Relayhost benutzen darf, um Emails nicht nur lokal, sondern auch nach draußen zu senden. Anmerk.: Ich nutze dieses Feature nicht, finde es aber so interessant, dass ich es trotzdem hier beachte.
smtpd_recipient_restrictions	Prüft abschließend, ob eine Mail bearbeitet oder verworfen wird.

Warum nur diese drei? Weil es genau die sind, die ich vor dem Hintergrund, dass man das eigentlich gar nicht braucht, dennoch als Verbesserung betrachte. Warum überhaupt? Die Frage ist einfach zu beantworteten. Beschränkungen wirken gegen Missbrauch, sie verhindern unberechtigte Benutzer oder unberechtigte Benutzung und wirken beispielsweise auch gegen Mail-Bots, die unseren Mailserver als Mailschleuder für Spam verwenden wollen. Das letztere kann nur natürlich nur passieren, wenn unser Mailserver auch aus dem Internet erreichbar ist.

Die Restrictions führen bei einem Ereignis (siehe oben in der Tabelle "*vor*" und "*nach*") eine Prüfung auf gewisse Merkmale einer Mail durch. Die zeitliche Besonderheit ist, dass bei den Prüfungen die Mail noch gar nicht vollständig vorliegen muss. Was das bedeutet, kann man am besten verstehen, wenn man später einmal eine Mail via openssl mit Einzelanweisungen von Hand gesendet hat, welches z.B. die folgenden Schritte erfordert:

1. Verbinden mit Host und auf Bestätigung warten
2. "*helo*" resp „ehlo“ (für TLS) senden und auf Bestätigung warten
3. "*mail from*" senden und auf Bestätigung warten
4. "*rcpt to*" senden und auf Bestätigung warten
5. "*data*" senden und auf Bestätigung warten

Bei der Verbindung zu Postfix über Openssl wird eine Codierung von Anmeldename und Password im Base64-Format erwartet:

```
$ echo -n 'thomas@mail' | base64
dGhvbWFzQG1haWw=

$ echo -n 'Mailsys-rpi3.toms_passwd' | base64
TWFpbHN5cy1ycGkzLnRvbXNfcGFzc3dk
```

Jeweils passend zum zeitlichen Eintreten eines Ereignisses wirkt die entsprechende Regel:

```
$ openssl s_client -starttls smtp -connect 10.10.1.2:25
CONNECTED(00000003)                       <---- smtpd_client_restrictions
::::::
250 mail
    EHLO TPC                              <---- smtpd_helo_restrictions

Local-Server
250-mail
250-PIPELINING
250-SIZE 10240000
250-ETRN
250-AUTH PLAIN LOGIN
250-ENHANCEDSTATUSCODES
250-8BITMIME
250-DSN
250-SMTPUTF8
250 CHUNKING

    auth login dGhvbWFzQG1haWw=
334 UGFzc3dvcmQ6
    TWFpbHN5cy1ycGkzLnRvbXNfcGFzc3dk
235 2.7.0 Authentication successful

    mail from:<thomas@mail>               <---- smtpd_sender_restrictions
250 2.1.0 Ok                              <---- smtpd_relay_restrictions

    rcpt to:<silvia@mail>
250 2.1.5 Ok                              <---- smtpd_recipient_restrictions
```

```
data

To: silvia@mail
From: thomas@mail
Subject: Hey my first email                  Als kompletten Block per Copy/Paste
                                             einfügen
This is my first email on debian
postfix after installing and
configuring it. It was easy

crlf . crlf                                  <---- enter-dot-enter = mail senden

250 2.0.0 queued as L05a53t47CcWm6Z          <---- smtpd_data_restrictions
    Q
DONE
```

Eine Restrictions-Regel hat 3 mögliche Rückgaben für das Prüfungsergebnis dieser Regel. Hier ein Beispiel für die smtpd_sender_restrictions, deren primäres Ergebnis entsprechend der zwei Einträge entwender permit oder reject ist, je nachdem, wer der Absender war.

thomas.addams_t@mail	permit	Diesem Absender ist es erlaubt, eine Mail zu senden
manuel.addams_g@mail	reject	Diesem Absender ist es verboten, eine Mail zu senden

Die dritte Möglichkeit ist dunno, die dann eintritt, wenn keine der angegebenen Regeln passt. Der Absender Thomas ist erlaubt, Manuel ist verboten, für Silvia ist jedoch keine Regel hinterlegt, also würde eine Mail von Silvia zum Ergebnis dunno führen. Am besten lässt sich dunno mit *"Don't know"* übersetzen. Und sofern nicht ein eindeutiges Ergebnis ermittelt werden konnte, wird Silvias Mail folgerichtig und zur Sicherheit korrekt transportiert.

Dieses Vorgehen passt zur vorherigen Aussage, dass Postfix eigentlich keine Restrictions braucht. Wenn also entweder gar keine Regeln definiert sind oder mit keiner Regel eine Übereinstimmung bei den Prüffaktoren für einen Abbruch gefunden wurde, wird mit dem Prüfungsergebnis dunno im übertragenen Sinne ein Default-Permit angenommen und die Mail wird ordnungsgemäß transportiert. Das verhindert, dass durch Fehler im Regelwerk oder durch ein vielleicht zu restriktives Regelwerk Emails fälschlicherweise abgewiesen werden.

Der wirkliche Sinn von permit und reject liegt meiner Meinung nach jedoch in Black- oder Whitelists, und zwar wenn man eine der zwei folgenden Regeln anwenden will:

- Es ist grundsätzlich alles verboten, was nicht ausdrücklich erlaubt ist = Whitelist = namentliche permits
- Es ist grundsätzlich alles erlaubt, was nicht ausdrücklich verboten ist = Blacklist = namentliche rejects

Was ist aber mit den Fällen, wenn man nur Ausnahmen behandeln will, aber ansonsten weder ausdrücklich erlauben noch ausdrücklich verbieten will? Genau dieses Problem löst dunno.

Wie man oben im Beispiel der manuellen durchgeführten Verbindung sieht, werden die Prüfungen sukzessive bei Eintreten eines Ereignisses durchgeführt. Und wie man jetzt unschwer erraten kann, liegen speziell hier in diesem Beispiel wegen der manuellen Eingabe an der Tastatur durchaus mehrere Sekunden zwischen den Prüfungen - alle Datenfelder werden ja einzeln an der Tastatur getippt. Dieses Verständnis ist wichtig für das Verstehen der Tatsache, dass sowieso keines der Prüfungsergebnisse zu einer sofortigen abschließenden Bearbeitung eine Mail führt. Das bedeutet, ein permit als Ergebnis der ersten Prüfung in der Regel smtpd_client_restrictions führt NICHT zum sofortigen Versenden der Email und ein reject führt NICHT zum sofortigen Abbruch des gesamten Prozesses - das betrifft alle Regeln bis einschließlich zu smtpd_recipient_restrictions.

Was bewirken also permit und dunno? Man könnte jetzt ganz lapidar sagen *"die tun eigentlich gar nix"*, permit und dunno haben faktisch die gleiche Wirkung, als wäre gar keine Regel gesetzt - die Mail wird gesendet. Der praktische Nutzen von permit ist, dass eine sehr umfangreiche Regel vielleicht nicht hoch zeitaufwendig bis ganz zum Ende abgearbeitet werden muss. Das bedeutet, wenn *'ich'* sende und früh am Anfang der Regel wird festgestellt, dass *'ich'* senden darf, ist es nicht notwendig, die ganze Regel noch weiter bis zum Ende zu bearbeiten und unnötigerweise alle anderen möglichen Einträge zu prüfen.

Mit permit wird die Regel sofort beendet und der allgemeine Prozess wird nach dieser Regel fortgesetzt. Das kann durchaus eine gewichtige Rolle spielen, wenn im gewerblichen Umfeld hunderte oder tausende Mails innerhalb einer Minute bearbeitet werden müssen, unter solchen Umständen sollte die Bearbeitungszeit von Regeln so kurz wie möglich sein. War das Ergebnis hingegen dunno, wurde die Regel komplett durchgearbeitet und es wurde keine Übereinstimmung gefunden, bzw. es konnte weder Erlaubnis noch Abbruch festgestellt werden.

Was bewirkt reject? Zunächst mal natürlich auch die sofortige Beendigung der weiteren Bearbeitung dieser Regel - aus dem gleichen Grund der Prozess-Beschleunigung wie zuvor. Welchen Sinn hat es jetzt noch alle anderen möglichen Absender zu prüfen, wenn doch schon festgestellt wurde, dass diese Aktion nicht erlaubt ist? Die Regel wird also beendet und der allgemeine Prozess nach dieser Regel wird fortgesetzt. Häh ...?... wie fortgesetzt..?... ist das nicht ein Irrtum ...?... das Ergebnis war doch reject. Nein, kein Irrtum, der Prozess wird auch nach einem frühen reject trotzdem immer komplett bis zum Erreichen der Regel smtpd_recipient_restrictions durchgeführt. Und dafür bestehen wirklich gute technische Gründe.

Es gibt unzählige Mail-Client-Programme, die großen und allgemein bekannten sind zweifelsfrei Outlook und Thunderbird. Daneben gibt's aber auch noch Clients wie z.B. Evolution und Claws-Mail, eine große Anzahl an Mail-Apps für Smartphones und Tablets, und diese jeweils für Android und Apple. Und es besteht bei all diesen Clients nicht wirklich Einigkeit darüber, wie mit einem Abbruch umzugehen ist. Theoretisch ist es sogar denkbar, dass eine Client-Anwendung bei einem frühen Abbruch z.B. durch smtpd_client_restrictions denkt *"oh, ein technischer Fehler oder Leitungsfehler, ich muss es noch mal versuchen"*. Und das würde vielleicht dazu führen, dass die Anwendung es versucht und versucht und versucht und versucht und versucht und versucht usw. usw. - und jedes mal verweigert Postfix erneut wegen des reject an immer gleicher Stelle die weitere Bearbeitung der Mail.

Um dieses Problem zu umgehen, wird der Prozess also immer fortgesetzt bis "rcpt to" vorliegt und quasi smtpd_recipient_restrictions endültig entscheidet. Bei einem vorliegenden reject wird der Vorgang erst jetzt endgültig angebrochen. Das ist dann der Punkt, an dem auch alle Mail-Clients eindeutig mit einem Abbruch umgehen können. Faktisch bedeutet das, ein frühes reject in einer Regel beendet nicht sofort den Prozess als solches, sondern setzt nur einen Marker, der letztendlich dazu führt, dass der Prozess nach "rcpt to" tatsächlich abgebrochen wird. Also reicht ein einziges reject in irgendeiner der Regeln aus, um eine Mail schlussendlich abzuweisen. Die zuvor vielleicht ermittelten permit und dunno haben an diesem Punkt faktisch keine Bewandtnis mehr.

Um das etwas besser der menschlichen Denkweise anzupassen, könnte man die Kernaussage von Postfix sinngemäß sprechend so übersetzen: *"...ich gehe generell davon aus, dass die Mail erlaubt ist und prüfe nur, ob es Umstände für ein Verbot gibt, welches ich mir erst mal nur merke und erst am Ende anwende. Sowohl permit als auch reject verhindern, dass ich unnötig weitere lange Zeit mit der Bearbeitung einer Regel verschwende, deren Ergebnis ich schon kenne."*

Sehen wir uns nun an, wie die 3 folgenden und zuvor genannten Regeln implementiert sind:

check_sender_filter legt fest, welche lokalen Absender-Adressen berechtigt sind, eine Email zu versenden. Alle unberechtigten Absender werden abgewiesen. Das bedeutet, es ist nicht möglich eine quasi anonymisierte und unbekannte Fake-Absender-Adresse zu verwenden, um eine missbräuchliche Email zu versenden. Hier ist es eine White-List, alle andere Sender sind verboten.

```
root@raspi3:/etc/postfix
# ne /etc/postfix/rules/check_sender_filter
```

```
# Authorized or forbidden senders to send Email
# default in main.cf = reject
#================================================

root@mail                   permit

thomas.addams_t@mail        permit
thomas.addams_g@mail        permit
tom.addams@mail             permit

silvia.addams_t@mail        permit
silvia.addams_g@mail        permit

manuel.addams_t@mail        permit
manuel.addams_g@mail        permit

familie@mail                permit
```

check_sender_relayaccess ermittelt, ob der Absender einen Relayhost verwenden darf. Hier in diesem Beispiel könnte Manuel also nur interne Mails innerhalb des LAN's versenden, Emails über das Internet an Empfänger außerhalb des lokalen Netzwerks sind verboten. Hier ist es eine Blacklist, alle anderen sind erlaubt.

```
root@raspi3:/etc/postfix
# ne /etc/postfix/rules/check_sender_relayaccess
```

```
# Authorized or forbidden senders to send via relayhost
# default in main.cf = permit/dunno
#=======================================================

manuel.addams_t@mail        reject
manuel.addams_g@mail        reject
```

Für die recipient-restrictions ist keine eigene Regeldatei hinterlegt, stattdessen sind nur Postfix-Standard-Prüfmethoden in der main.cf veranlasst. Hier ist es aber wichtig zu sehen, dass diese Regel als Default-Ergebnis permit zurückgibt. Wenn also nicht vorher eine ausdrückliche Abbruchbedingung festgestellt wurde, wird eine zu versendende Email ordnungsgemäß transporiert.

```
smtpd_recipient_restrictions =
    permit_mynetworks
    reject_non_fqdn_sender
    reject_non_fqdn_recipient
    reject_unauth_destination
    permit
```

Wenn in den vorherigen Prüfungen keine Abbruchbedingung ermittelt wurde und die Email nun tatsächlich an einen entfernten Empfänger über das Internet transportiert werden soll, braucht es nun für den erfolgreichen Abschluss dieses Prozesses noch die folgenden zum Absender gehörenden Anmelde-Daten, die wie zuvor auch über mehrere Regeldateien definiert sind.

get_sender_relayhost ermittelt vor dem tatsächlichen Sendevorgang zur lokalen Absender-Adresse den für diesen Sender zu verwendenden SMTP-Server als Smarthost, passend zu den Anmeldedaten in get_relayhost_passwd

```
root@raspi3:/etc/postfix
# ne /etc/postfix/rules/get_sender_relayhost
```

```
# Per-sender provider; see also /etc/postfix/get_relayhost_passwd
#=================================================================

root@mail                  smtp.toml.de

thomas.addams_t@mail       smtp.toml.de
thomas.addams_g@mail       mail.gmx.net
tom.addams@mail            smtp.gmail.com

silvia.addams_t@mail       smtp.toml.de
silvia.addams_g@mail       mail.gmx.net

manuel.addams_t@mail       smtp.toml.de
manuel.addams_g@mail       mail.gmx.net

familie@mail               smtp.toml.de
```

get_relayhost_passwd ist eine Regel, mit der vor dem tatsächlichen Sendevorgang zur lokalen Absender-Adresse die Anmeldedaten zum SMTP-Server des Email-ISP ermittelt werden. Postfix muss sich zum Senden der Email genau so beim Mail-ISP anmelden und authentifizieren, wie wir das tun müssten, wenn wir uns z.B. über einen Browser auf der Web-Seite des Postfachs anmelden.

```
root@raspi3:/etc/postfix
# ne /etc/postfix/rules/get_relayhost_passwd
```

```
# Per-sender authentication; see also /etc/postfix/get_sender_relayhost
#=======================================================================

root@.mail                 thomas.addams@toml.de:irgendeingeheimespwd

thomas.addams_t@mail       thomas.addams@toml.de:irgendeingeheimespwd
thomas.addams_g@mail       thomas.addams@gmx.de:irgendeinanderesgeheimespwd
tom.addams@mail            tom.addams@gmail.com:auchganzgeheim

silvia.addams_t@mail       silvia.addams@toml.de:eineigenespwd
silvia.addams_g@mail       silvia.addams@gmx.de:einandereseigenesgeheimespwd

manuel.addams_t@mail       manuel.addams@toml.de:manuelspwd
manuel.addams_g@mail       manuel.addams@gmx.de:manuels2pwd

familie@mail               thomas+silvia.addams@_toml.de:familypassword
```

remove_header_sensitives entfernt zusätzliche Informationen aus dem Mail-Header, die weder für den Inhalt der Mail noch für den Transport relevant sind. Es ist durchaus möglich, dass solche Felder auch sensible und datenschutzrelevante Informationen beinhalten können, also ist es sinnvoll, solche Inhalte zu entfernen. Das ist jetzt hier einer der wenigen Punkte, bei denen es mir schwerfällt, dass auch wirklich zu verifizieren. Wie provoziere ich z.B. relevante Feldinhalte in die Mail hinein, um dann zu prüfen, dass sie ohne Filter gesendet werden und mit Filter entfernt wurden? Eine schwierige Frage, bei der ich nun darauf vertraue, dass der Filter einfach nur das das tut, was er verspricht zu tun… allerdings ist zu bemerken, der Filter selber tut ja gar nichts…. es ist postfix, welchen diesen Filter aktiv anwendet.

```
root@raspi3:/etc/postfix
# ne /etc/postfix/rules/remove_header_sensitives
```

```
# Remove sensitive information from email headers with postfix
#==============================================================

/^Received:.*with ESMTPSA/              IGNORE
/^X-Originating-IP:/                    IGNORE
/^X-Mailer:/                            IGNORE
/^Mime-Version:/                        IGNORE
/^From:.*(root)/                        REPLACE From: <email_address>
```

Als letztes die eigentlichen Konfigurationseinstellungen für Postfix:

```
# touch /etc/postfix/main.cf
# chown root:postfix /etc/postfix/main.cf; chmod 640 /etc/postfix/main.cf
```

Rechteeinstellung: -rw- r-- --- root:postfix main.cf

```
root@raspi3:/etc/postfix
# ne /etc/postfix/main.cf
```

```
# Version: 5.4
# Date:    25.07.2022
#============================================================================
# mydomain      = externer (www) domain-name des Mailservers (domain.tld)
# myhostname    = externer (www) hostname des Mailservers    (host.domain.tld)
# myorigin      = interner (lan) hostname/domain-name des Mailservers
# mydestination = erlaubte Domains für Annahme zu sendender Mails
# mynetworks    = vertrauenswürdige Netzwerke

mydomain        = mail
myhostname      = mail
myorigin        = mail
mydestination   = $myhostname, $myhostname.localdomain, localhost, localhost.localdomain
mynetworks      = 127.0.0.0/8 [::1]/128 10.0.1.0/24 10.0.8.0/23

#--------------------------------------------------------------------------

# disable backwards compatibility
compatibility_level = 3.6

biff = no
append_dot_mydomain = no
readme_directory = no

mailbox_command =/usr/lib/dovecot/deliver
mailbox_size_limit = 0
recipient_delimiter = +
inet_interfaces = all
inet_protocols = ipv4
disable_vrfy_command = yes

#--------------------------------------------------------------------------
# Redirect lokal adressierte Emails direkt zum Dovecot-Empfänger

transport_maps = hash:/etc/postfix/check_local_transport_maps
virtual_transport = lmtp:unix:private/dovecot-lmtp
virtual_alias_maps = hash:/etc/postfix/check_local_alias_maps

#--------------------------------------------------------------------------
# Lokal SMTP-Daemon, TLS-Settings

smtpd_tls_security_level = encrypt
smtpd_tls_auth_only = yes
smtpd_tls_ciphers = high
smtpd_tls_protocols = TLSv1.3 TLSv1.2, !TLSv1.1, !TLSv1, !SSLv2, !SSLv3
smtpd_tls_exclude_ciphers = MD5, DES, ADH, RC4, PSD, SRP, 3DES, eNULL, aNULL
smtpd_tls_mandatory_protocols = TLSv1.3 TLSv1.2, !TLSv1.1, !TLSv1, !SSLv2, !SSLv3
smtpd_tls_mandatory_ciphers = high
smtpd_tls_mandatory_exclude_ciphers = MD5, DES, ADH, RC4, PSD, SRP, 3DES, eNULL, aNULL
smtpd_tls_received_header = yes
smtpd_tls_session_cache_timeout = 3600s
smtpd_tls_session_cache_database = btree:${data_directory}/smtpd_scache
smtpd_tls_loglevel = 1

smtpd_tls_CApath            = /etc/postfix/certs
smtpd_tls_CAfile            = /etc/postfix/certs/cacert.pem
smtpd_tls_cert_file         = /etc/postfix/certs/servercert.pem
smtpd_tls_key_file          = /etc/postfix/certs/serverkey.pem
smtpd_tls_dh1024_param_file = /etc/postfix/certs/dh4096.pem

smtpd_banner = $myhostname ESMTP $mail_name
smtpd_helo_required=yes

smtpd_sasl_security_options = noanonymous
smtpd_sasl_auth_enable = yes
smtpd_sasl_type = dovecot
smtpd_sasl_path = private/auth

tls_random_source = dev:/dev/urandom

#--------------------------------------------------------------------------
```

```
# Extended Restrictions

# 1. Black/Whitelist: Wer darf (nicht) senden? Default=reject!
smtpd_sender_restrictions =
    check_sender_access hash:/etc/postfix/check_sender_filter
    reject

# 2. Black/Whitelist: Wer darf (nicht) extern via Relayhost senden? (hier deaktiviert, default=dunno)
smtpd_relay_restrictions =
#   check_sender_access hash:/etc/postfix/check_sender_relayaccess
    permit_mynetworks
    reject_unauth_destination

# 3. Standard-Prüfmethoden. Default=permit!
smtpd_recipient_restrictions =
    permit_mynetworks
    reject_non_fqdn_sender
    reject_non_fqdn_recipient
    reject_unauth_destination
    permit

#-------------------------------------------------------------------------
# ISP SMTP, TLS-Settings

smtp_tls_note_starttls_offer = no
smtp_tls_security_level = encrypt
smtp_tls_ciphers = high
smtp_tls_protocols = TLSv1.3 TLSv1.2, !TLSv1.1, !TLSv1, !SSLv2, !SSLv3
smtp_tls_exclude_ciphers = MD5, DES, ADH, RC4, PSD, SRP, 3DES, eNULL, aNULL
smtp_tls_mandatory_protocols = TLSv1.3 TLSv1.2, !TLSv1.1, !TLSv1, !SSLv2, !SSLv3
smtp_tls_mandatory_exclude_ciphers = MD5, DES, ADH, RC4, PSD, SRP, 3DES, eNULL, aNULL
smtp_tls_mandatory_ciphers = high
smtp_tls_session_cache_database = btree:${data_directory}/smtp_scache

smtp_sasl_auth_enable = yes
smtp_sasl_security_options = noanonymous
smtp_sasl_password_maps = hash:/etc/postfix/get_relayhost_passwd

smtp_generic_maps = hash:/etc/postfix/check_generic_replace_from

smtp_sender_dependent_authentication = yes
sender_dependent_relayhost_maps = hash:/etc/postfix/get_sender_relayhost

#-------------------------------------------------------------------------
# Specifies which account sent mail to an alias

alias_maps = hash:/etc/aliases
alias_database = hash:/etc/aliases

#-------------------------------------------------------------------------
# Remove sensitive information from email-headers with postfix

mime_header_checks = regexp:/etc/postfix/remove_header_sensitives
header_checks = regexp:/etc/postfix/remove_header_sensitives

#===========================================================================
# Ende
```

Abschließend noch ein 3 Ergänzungen:

1. smtp_sasl_security_options = noplaintext noanonymous lautet eine Default-Einstellung von Postfix, die ich auf noanonymous geändert habe. Die Default-Einstellung bedeutet, dass Anmeldungen an den ISP-SMTP-Server nicht mit Plain-Text-Passwörtern erfolgen können, unverschlüsselte Passwörter werden also abgewiesen. Hierbei ist aber festzustellen, dass es eigentlich gar keine unverschlüsselten Passwörter gibt, denn der gesamte Datenverkehr ist schon durch die Einstellung smtp_tls_security_level = encrypt verschlüsselt.

   Die Option „may“ beschreibt ein ‚opportunistic TLS‘, damit ist zwar das Türchen „unverschlüsselt“ grundsätzlich noch geöffnet, mit encrypt ist der Traffic generell verschlüsselt, unverschlüsselter Verkehr ist nicht erlaubt. Wäre also das übertragene Password verschlüsselt, so würde ein verschlüsseltes Password innerhalb der verschlüsselten Datenübertragung stattfinden. Zum einen ist das unsinnig, zum zweiten lehnt beispielsweise Web.de eine doppelte Verschlüsselung einfach ab und verhindert die Anmeldung. Darüber hinaus können wir sowieso nicht beeinflussen, welche Arten der Datenübertragung die verschiedenen Mail-ISP überhaupt zulassen. Nur für die Anmeldung an unsere eigenen Services können wir definitive Vorgaben und ultimative Beschränkungen festlegen, was ich mit smtpd_tls_security_level = encrypt und smtpd_tls_auth_only = yes getan habe.

2. Die Parameter für mydomain, myhostname, myorigin sind im wesentlichen für öffentlich erreichbare Mailserver relevant, deren Name im Internet durch DNS aufgelöst werden kann bzw. werden muss. Dazu wäre natürlich zwingend eine korrekte FQDN-Syntax erforderlich. Hier wird jedoch nur eine lokale Installation eingerichtet, und deshalb war es hier ausreichend, den lokalen Namen 'mail' einfach im lokalen DNS einzutragen. Dieser symbolische Name wird zum Internet ausgehend ansonsten nur durch Postfix im HELO-Dialog bei der Verbindung zum Relayhost verwendet.

3. Die Einstellung mynetworks = 127.0.0.0/8 [::1]/128 10.0.1.0/24 10.0.8.0/23 zeigt, dass ich mehrere (hier 3) Netzwerke konfiguriert und damit für Sendevorgänge zugelassen habe. Siehe dazu auch den Artikel <Security>. Nur Clients aus einem dieser 3 Netze sind berechtigt Emails zu versenden. Warum 3 Netzwerke? Ganz einfach, ich nutze für die PCs zuhause unser statisches Heimnetzwerk, dazu für die mobilen Laptops und Smartphones ein mobiles Netzwerk via OpenVPN/TCP und ein weiteres mobiles Netzwerk via OpenVPN/UDP. Wenn für die Einrichtung des Mailservers nur das eigene statische Heimnetzwerk verwendet wird, wäre hier auch nur diese eine Netzwerk-Domain einzutragen.

Als letzte notwendige Aktion zur Einrichtung von Postfix wird das kleine Helfer-Script gestartet, welches in diesem Kapitel als erste Datei angelegt wurde.

```
root@raspi3:/etc
# cd /etc/postfix/rules/
# ./refresh-rules
```

Sofern beim Aktualisieren der Regeldateien und durch das im Script enthaltene *postfix check* keine Fehler entdeckt werden, kann die Frage nach einem Neustart des Mailsystems mit „y“ beantwortet werden. Wenn Fehler berichtet werden, müssen diese Fehler natürlich zuerst behoben werden.

Zur Kontrolle werfen wir einen kurzen Blick auf die erstellten Dateien:

```
root@raspi3:/etc
ls /etc/postfix/*.db
```

Und ab jetzt sollte es auch problemlos möglich sein, dass auf Seite 43-44 beschriebene Openssl-Connect-Beispiel im Command-Line-Modus durchzuspielen.

13. Getmail – Einrichtung des Mail-Retrival-Agents

In diesem Kapitel wird Getmail eingerichtet, welches aufgrund der von mir festgelegten Rahmenbedingungen schon eine kleine Herausforderung und einigen Aufwand darstellt. Getmail ist das Programm, welches die relevanten Postfächer bei den Mail-ISP im Internet abfragt und die dort empfangenen Mails auf unseren eigenen Mailserver überträgt. Wir erinnern uns, ganz zu Anfang hatte ich festgelegt, dass mein Mailserver aus Gründen der Datensicherheit und der Notwendigkeit eines absolut integren Systemzustandes nicht direkt vom Internet erreichbar sein darf. Aber wegen genau dieser Vorgabe ist mir jedoch der allgemein übliche Weg beim Downloaden der Emails verwehrt und es ist stattdessen eine besondere Lösung notwendig.

Der üblicherweise praktizierte Weg ist, das ein Mailserver alle relevanten ISP-Postfächer in einem vorgegebenen Zyklus automatisch abfragt und die dort vorhandenen Emails in die lokalen Postfächer überträgt. Das bedeutet, unser Mailserver enthält in diesem Fall immer sehr zeitnah alle aktuell eingegangenen Emails. Wie aktuell er tatsächlich ist, ist natürlich durch die Wartezeit des vorgegebenen Zyklus festgelegt. Anfangs hatte ich mir mal überlegt, die automatische Abfrage alle 15 Minuten durchzuführen… was meiner Meinung wirklich ausreichend aktuell bedeutet. Aber genau diese Verfahrensweise hätte bei der Maßgabe *„unser Mailserver ist nicht übers Internet erreichbar“* die Folge, dass wir unterwegs auf dem Handy, Tablet oder Laptop praktisch keine Mails mehr einsehen können, wenn wir uns über „mobile Daten“ direkt beim Mail-ISP einloggen würden. Warum nicht? Weil die ISP-Postfächer quasi alle paar Minuten geleert werden und die Mails auf unseren Mailserver übertragen wurden… tja…das ist wirklich nicht gut.

Selbstverständlich ist es vorgesehen, sich auch von unterwegs via OpenVPN mit dem Mailserver zu verbinden. Aber der Aufwand ist um einiges größer und es dauert auch deutlich länger, weil jedes Mal eben zusätzlich auch noch die verschlüsselte Verbindung nach Hause aufgebaut werden muss. Wenn man allerdings gewohnt ist, mal eben schnell auf dem Smartphone „*k9mail*“ zu starten und „*synchronisieren*“ anzutippen, so würde das nicht mehr funktionieren. Technisch betrachtet läuft natürlich alles problemlos ab, es werden nur nie irgendwelche Emails gefunden – eben weil unser Mailserver sie vermutlich kurz vorher selber runtergeladen hat. Der Zugriff auf unseren Mailserver durch die Client-Geräte ist via OpenVPN natürlich vorhanden, aber dieser Zugang ist nach meiner Vorstellung eher dafür gedacht, z.B. auf Reisen die Emails ganz entspannt einmal am Tag über eine sichere Leitung abzurufen, wenn man für sicheres Surfen sowieso den heimischen Router verwendet und sowieso einen Tunnel etabliert hat.

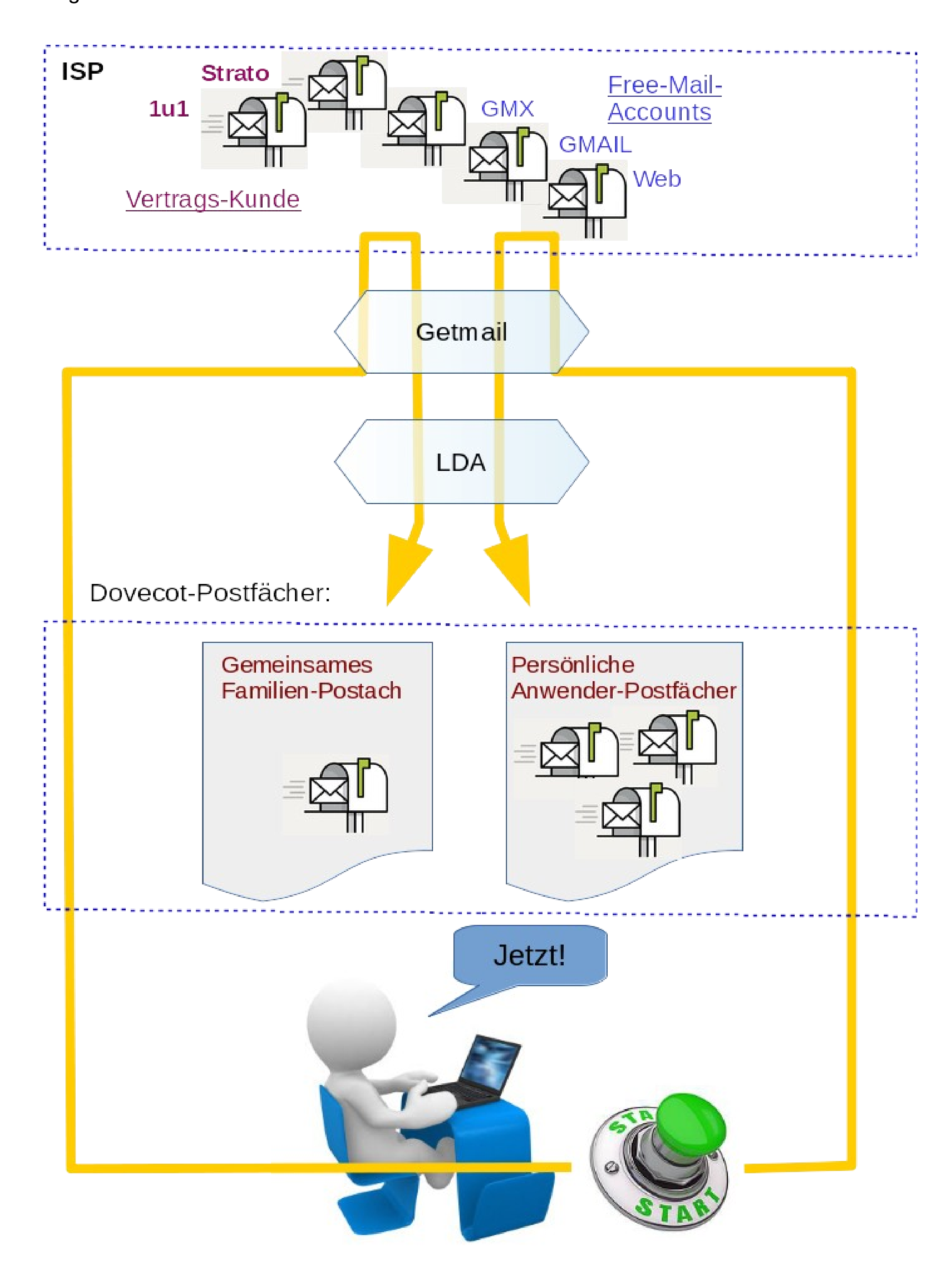
Wie ich anfangs in dieser Anleitung schon festgestellt habe, ist es selbstverständlich auch keine Alternative, die Daten einfach vollständig bei den Mail-ISP zu belassen und grundsätzlich via IMAP zu bearbeiten. Eine wesentliche Prämisse bei diesem Vorhaben war ja, keine dauerhaft anwachsende Datensammlung auf dem Server eines fremden Unternehmens zu ermöglichen, dessen Art und Weise des Umgangs mit unseren privaten Daten ich nicht kenne. Das runterladen der Emails ist also obligatorisch.

Nun muss das folgende Problem gelöst werden:

Unser Mailserver darf die in Frage kommenden Anwender-Postfächer bei den Mail-ISP nicht zyklisch übertragen, weil außerhalb unseres Netzwerkes neue Emails im normalen Berufs-Alltag dann nicht mehr ohne Aufwand mit den Mobil-Geräten geöffnet werden könnten. Die endgültige Übertragung der Emails darf erst als Reaktion des Servers erfolgen, wenn sich ein User auf seinem Client-Gerät im lokalen Netzwerk z.B. in Thunderbird anmeldet.



Zur Lösung dieses Problems habe ich mich entschieden, für die Behandlung der Anwender-Anmeldung in Thunderbird einen Eventhandler auf dem Server einzusetzen, der genau dann die Postfächer dieses Anwenders abfragt, wenn er seinen Thunderbird startet. Schauen wir uns zum besseren Verständnis einfach die folgende Grafik an.

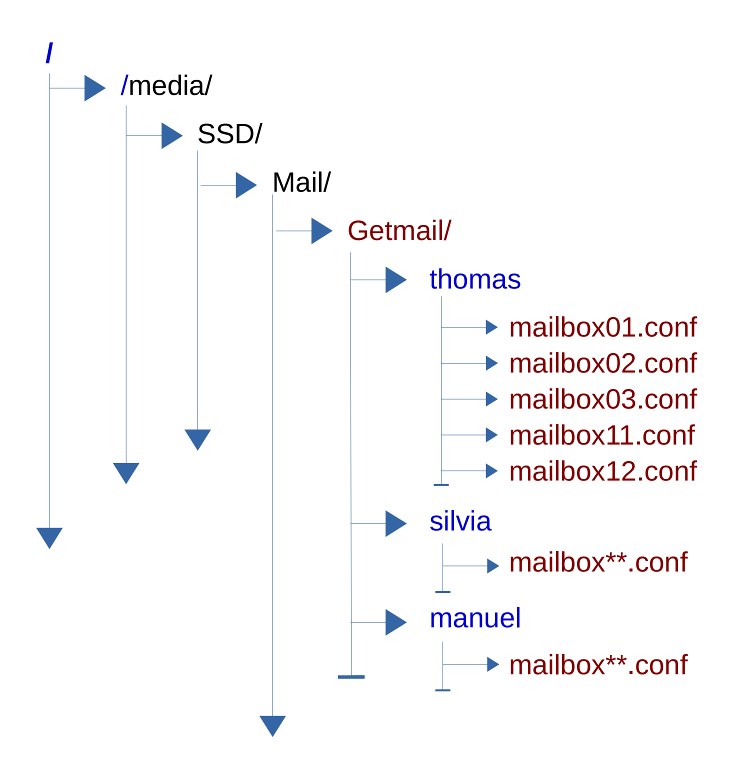


Die Anwender geben mit ihrer Anmeldung auf dem Dovecot-Server (das passiert automatisch beim Start von Thunderbird) selber das Startsignal für den Download sowohl für ihre eigenen Emails als auch für das gemeinsame Familien-Mailkonto. Nach dem Ereignis "Anmeldung" im Dovecot-Email-Konto erfolgt über die interne Dovecot-Funktion Post-login-Scripting der Start des zuvor schon genannten Getmail-Eventhandlers.

Beginnen wir nun mit den einfachen Aufgaben, und zwar den Einstellungsdateien, über die Getmail die Postfächer abfragt. Den weiter oben angesprochenen Eventhandler stelle ich hier zunächst noch einmal zurück, weil sich das Problem erst dann stellt, wenn alle Programme grundsätzlich fertig eingerichtet sind und ordentlich ihren Job verrichten. Auch hier wieder zur Erinnerung noch mal der wichtige Hinweis, dass die notwendigen Dateien auf der SSD liegen, und zwar im Verzeichnis

/media/SSD/Mail/Imap/Getmail

Für jeden User hatten wir zuvor im Getmail-Verzeichnis ein User-Verzeichnis angelegt (siehe Seite 16) und dort wird nun für jedes abzufragende ISP-Postfach eine eigene Conf für den Getmail-Prozess angelegt. Das ermöglicht mir über ein Script nur die für den aktuell angemeldeten User betroffenen Postfächer isoliert zu bearbeiten, ohne die Postfächer der andere User zu beachten.



Für jedes der vorhandenen ISP-Postfächer wird nun eine eigene conf-Datei entsprechend der zwei folgenden Beispiele angelegt. Allerdings nur für diejenigen Postfächer, die ich als „seriös“ bezeichnet habe und die mit eigenem Real-Namen personifiziert sind. Für die Postfächer, deren Email-Adressen ich anfangs als Fake-Adressen bezeichnet habe, sind keine Conf-Dateien notwendig. Diese Fake-Adressen werden nicht abgeholt, sondern von allen Geräten nur via IMAP gehandhabt.

Anfangs habe ich eine Zeitlang überlegt, ob es nicht auch eine elegante Lösung wäre, diese Conf-Dateien jeweils bei Bedarf dynamisch zu generieren. Das wäre problemlos möglich, da alle Infos sowieso schon in den Postfix-Files hinterlegt sind und hier erneut verwendet werden könnten - aber dann hat sich der Gedanken durchgesetzt, dass das Erstellen individueller Files tatsächlich nur ein einmaliger Aufwand ist und danach vermutlich sehr lange Bestand hat. Und wenn sich mal ein Password irgendwo ändert, ist das dann auch schnell erledigt. Also… was solls…?… bleib ich einfach bei den statischen Conf-Files.

Die rot geschriebenen Parameter müssen natürlich auf die tatsächlichen Anmeldedaten der User geändert werden. Zusätzlich ist zu beachten, dass es keine eindeutige Regel für eine immer gleiche Schreibweise oder Bezeichnung der verschiedenen POP3-Server gibt. Das heißt, der FQDN eines ISP-POP3-Server wird vom Provider vorgegeben. Diese Information ist aber bei allen Mail-ISP auf deren Web-Sites zu finden.

```
root@raspi3:/media/SSD/Mail
# ne /media/SSD/Mail/Getmail/thomas/mailbox01.conf
```

```
[options]
verbose = 0
delete = true
read_all = true
message_log = /media/SSD/Mail/Getmail/thomas/thomas_addams_at_toml_de.log

[retriever]
type = SimplePOP3SSLRetriever
server = pop3.toml.de
username = thomas.addams@toml.de
password = irgendeingeheimespwd

[destination]
type = MDA_external
path = /usr/lib/dovecot/deliver
arguments = ("-d","thomas.addams_t")
```

```
root@raspi3:/media/SSD/Mail
# ne /media/SSD/Mail/Getmail/thomas/mailbox02.conf
```

```
[options]
verbose = 0
delete = true
read_all = true
message_log = /media/SSD/Mail/Getmail/thomas/thomas_addams_at_gmx_de.log

[retriever]
type = SimplePOP3SSLRetriever
server = pop.gmx.net
username = thomas.addams@gmx.de
password = irgendeinanderesgeheimespwd

[destination]
type = MDA_external
path = /usr/lib/dovecot/deliver
arguments = ("-d","thomas.addams_g")
```

Wenn alle Conf-Files für alle Mail-User erstellt sind, müssen unbedingt noch die Rechte korrigiert werden. Zum einen, um sicherzustellen, dass die Prozesse, die der User unter seiner UID anstößt, diese Dateien überhaupt lesen dürfen. Zum anderen, um sicherzustellen, dass auch wirklich niemand anderes als nur die Prozesse des Mailservers diese Daten lesen darf. Schließlich sind in diesen Dateien ja die tatsächlichen Anmeldedaten der einzelnen Anwender in ihre jeweiligen ISP-Mail-Postfächer hinterlegt.

```
# find /media/SSD/Mail/Getmail/ -exec chown vmail:vmail {} +
# find /media/SSD/Mail/Getmail/ -iname "*.conf" -type f -exec chmod 600 {} +
```

Als letztes lassen wir uns zur Kontrolle einmal die erstellten Dateien anzeigen und prüfen, ob UID und GID soweit korrekt gesetzt sind:

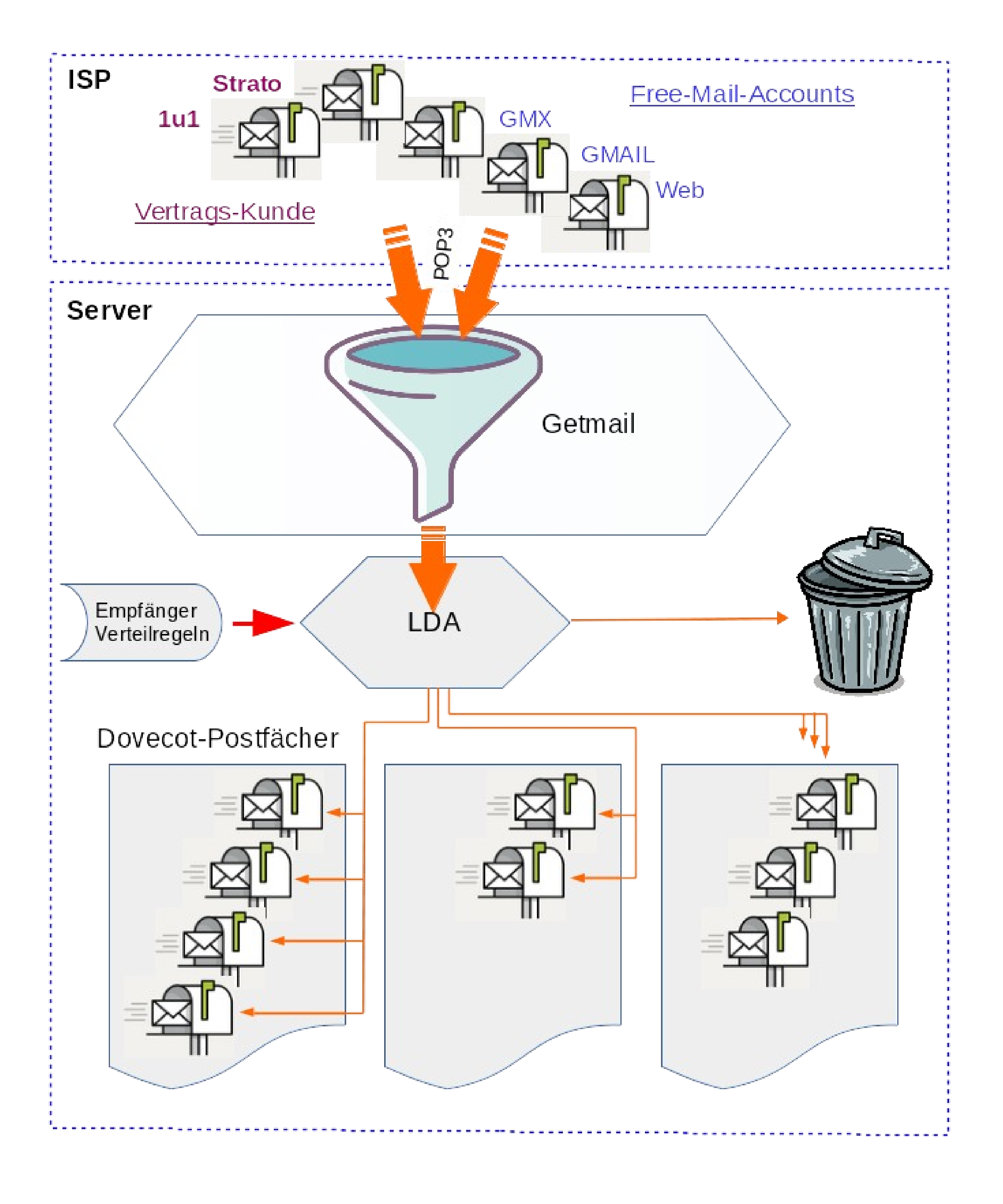
```
find /media/SSD/Mail/Getmail -iname "*.conf" -type f -ls
```

Ein weiterer Blick zur Kontrolle in die Verzeichnisse sollte nun zeigen, dass jetzt überall nur der User vmail als Eigentümer gesetzt ist und ‚Group‘ und ‚Other‘ keinerlei Rechte haben, darüber hinaus Log-Pfad und Usernamen korrekt sind:

```
find /media/SSD/Mail/Getmail -iname "*.conf" -type f -exec grep Getmail {} +
find /media/SSD/Mail/Getmail -iname "*.conf" -type f -exec grep "username" {} +
```

14. Sieve – Einrichtung der Filterregeln für die Zustellung durch den LDA

Die Aufgabe des Dovecot LDA und der Sieve-Filterregeln ist einfach und der Auftrag lautet: Nehme alle Emails an, die Getmail von den Mail-ISP abruft und führe die Zustellung in das jeweilige Postfach des Anwenders nach entsprechenden Regeln aus. Der Local Delivery Agent (LDA) übernimmt also die Zustellung der Email an das entsprechende Postfach eines Users. Welches Postfach das ist, hatten wir ja zuvor in den Getmail-Confs festgelegt. Der Sieve-Filter prüft darüber hinaus, ob eine Mail bestimmte Voraussetzungen erfüllt, um diese Default-Bearbeitung zu verlassen und einen anderen Weg zu nehmen, zum Beispiel die Email direkt in den Spam-Ordner des Postfachs zu verschieben, oder sie wird einfach gedropt. Schauen wir also noch mal kurz auf den Datenfluss - hier an dieser Stelle geht es speziell um unsere Sieve-Filterregeln:



Das Prinzip ist hier relativ einfach. Jedes Postfach kann eine eigene Regeldatei haben, mit der die eingehende Post für dieses Postfach geprüft wird. Sofern ein Postfach keine Regeldatei hat, wird die im übergeordneten Verzeichnis liegende Default-Regel angewandt.

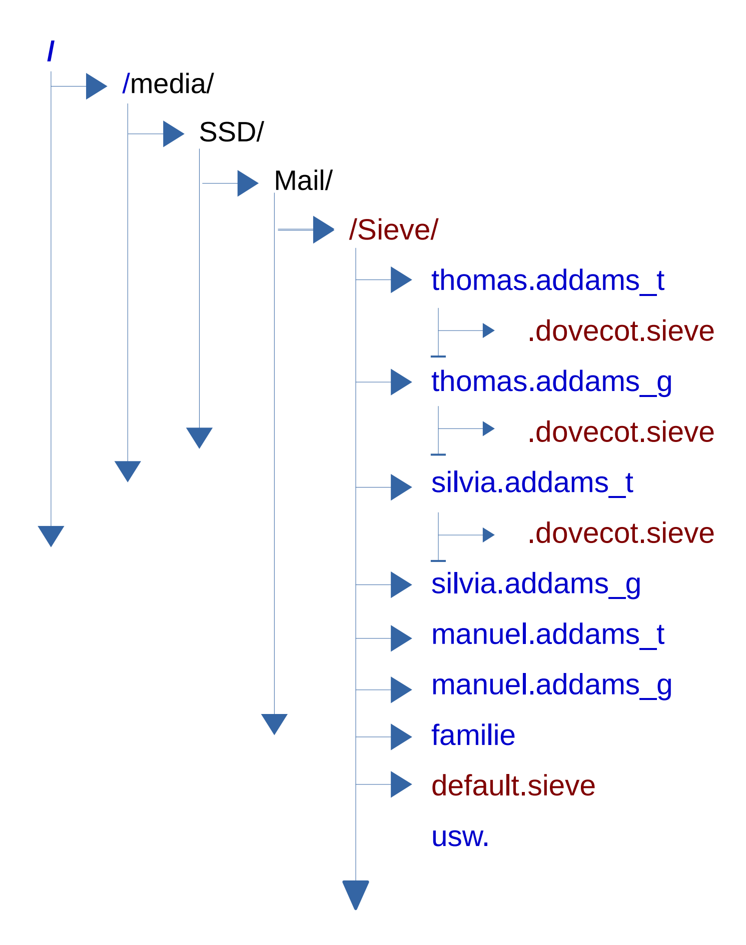
Die beiden Dateinamen dovecot.sieve und default.sieve sind hier nicht willkürlich gewählt, sondern sind durch unsere eigene Konfiguration vorgegeben. Die Dovecot.conf enthält dazu die hier relevanten zwei Zeilen (siehe Seite 24 + 25):

```
sieve = /media/SSD/Mail/Sieve/%n/.dovecot.sieve
sieve_default = /media/SSD/Mail/Sieve/default.sieve
```

.dovecot.sieve wird aber auch genau so im Dovecot-Wiki genannt. Insbesondere Im Zusammenhang mit „Managed Sieve“ ist diese Datei dann keine wirkliche Datei mehr, sondern ein Symlink auf die vom User selber über sein Email-Client-Programm (z.B. Thunderbird) erstellte persönliche Regeldatei.

Weil ich allerdings zunächst kein „Managed Sieve“-Protokoll (wäre notwendig für die Kommunikation zwischen Email-Client und Server) vorgesehen habe, kann .dovecot.sieve der Einfachheit halber auch eine echte Datei sein und nicht ein Symlink auf eine andere Datei. Ganz wichtig ist an dieser Stelle jedoch der Hinweis, dass der vorangestellte Punkt im Dateinamen kein Fehler, sondern regulärer Bestandteil des Namens ist. Der vorangestellte Punkt hat Linuxtypisch die Auswirkung, dass diese Datei mit den Standard-Einstellungen der meisten Filemanager als Hidden-File nicht angezeigt wird.

Die Regeldateien werden in der folgenden Verzeichnisstruktur angelegt. Die den Usern gehörenden Verzeichnisse enthalten hier nicht den Linux-Namen des Users, sondern den Postfach-Namen bzw. den Namen des virtuellen Dovecot-Users aus der Datei /etc/dovecot/passwd (siehe Seite 30).



Die jetzt jeweils im Dovecot-Homedir des virtuellen Dovecot-Users anzulegenden Sieve-Scripte bekommen alle die folgende Rechteeinstellung:

Dir	User	Gruppe	Alle		Rechte	Verzeichnis
-	rw	rw	---	660	vmail:vmail	/media/SSD/Mail/Sieve/$Postfach/*

Ich wiederhole das hier zur Sicherheit noch einmal: Bitte beim Erstellen der Dateien unbedingt auf die korrekte Schreibweise achten und den **Punkt** vor dem Dateinamen nicht vergessen oder außer acht lassen.

Ich habe hier folgend 3 unterschiedliche Beispiele aufgeführt, um wenigstens ansatzweise die hohe Flexibilität der Sieve-Filter-Scriptsprache und die grundsätzliche Syntax zu zeigen.

```
root@raspi3:/media
# cd /media/SSD/Mail/Sieve/thomas.addams_t
# touch .dovecot.sieve; ne .dovecot.sieve
```

```
require ["fileinto"];

if anyof (address :all :contains ["To", "Cc", "Bcc"] "thomas@mail")
{
    fileinto "INBOX";
}

elsif anyof (address :all :contains ["To", "Cc", "Bcc"] "thomas.addams@toml.de")
{
    fileinto "INBOX";
}

elsif header :contains "X-Spam-Flag" "YES"
{
    fileinto "Spam";
}

else
{
    fileinto "Spam";
}
```

Die Bedeutung des Inhalts der ersten Beispiel-Regel-Datei ist schnell erklärt:

1. Speichere lokale Post an thomas@mail in der Inbox des Postfachs des Dovecot-Postfach thomas.addams_t
2. Speichere eingehende Emails mit dem Empfänger (to, cc oder bcc) thomas.addams@toml.de wie zuvor in der Inbox
3. Speichere eingehende Emails im Ordner „Spam“, wenn das Spam-Flag mit Yes gesetzt ist.
4. Speichere alle Emails, die zwar in meinem ISP-Postfach empfangen wurden, aber in denen ich nicht ausdrücklich als Empfänger eingetragen bin *(typischerweise Spam oder andere unerwünschte Post)*, im Spam-Order des Dovecot-Postfachs thomas.addams_t

Eine besondere Variante, die ins User-Postfach zustellt und gleichzeitig eine Kopie archiviert:

```
# cd /media/SSD/Mail/Sieve/thomas.addams_g
# touch .dovecot.sieve; ne .dovecot.sieve
```

```
require ["copy", "fileinto"];

if anyof (NOT address :all :contains ["To", "Cc", "Bcc"] "thomas.addams@gmx.de")
{
    fileinto "Spam";
}
else
{
    redirect :copy "Mailarchiv@mail";
    fileinto "INBOX";
}
```

```
root@raspi3:/media
# cd /media/SSD/Mail/Sieve/silvia.addams_g
# touch .dovecot.sieve; ne .dovecot.sieve
```

```
require ["fileinto"];

if anyof (address :all :contains ["To", "Cc", "Bcc"] "silvia@mail")
{
    fileinto "INBOX";
}

elsif anyof (NOT address :all :contains ["To", "Cc", "Bcc"] "silvia.addams@toml.de")
{
    fileinto "Spam";
}

elsif header :contains "from" "Online-Versand" {
    discard;
}

elsif header :contains ["subject"] ["Finanzamt-Rechnung"] {
    discard;
}

elsif header :contains "X-Spam-Flag" "YES" {
  fileinto "Spam";
}

else
{
  fileinto "Spam";
}
```

Für alle weiteren noch existierenden Postfächer bzw. Email-Adressen, die ebenso übertragen werden sollen, können nun nach gleichem Muster einfach im jeweiligen Sieve-Ordner des Postfachs eigene Filter-Regeln eingerichtet werden, entweder angelehnt an eines der vorherigen Beispiele oder auch mit neuen eigenen Regeln. Nun fehlt noch die Default-Regel:

```
# cd /media/SSD/Mail/Sieve
# touch default.sieve; ne default.sieve
```

```
require "fileinto";

if header :contains "X-Spam-Flag" "YES" {
  fileinto "Spam";
}
```

Und am Ende einmalig die Rechte für alle neuen Dateien setzen und wieder die Abschluss-Kontrolle, ob uns Fehler unterlaufen sind.

```
# find /media/SSD/Mail/Sieve/ -iname "*.sieve" -type f -exec chown vmail:vmail {} +
# find /media/SSD/Mail/Sieve/ -iname "*.sieve" -type f -exec chmod 660 {} +
# find /media/SSD/Mail/Sieve/ -iname "*.sieve" -type f -ls
```

```
-rw-rw----  vmail  vmail  Jul 14 11:11 /media/SSD/Mail/Sieve/default.sieve

-rw-rw----  vmail  vmail  Jul 15 15:48 /media/SSD/Mail/Sieve/thomas.addams_t/.dovecot.sieve
-rw-rw----  vmail  vmail  Jul 14 15:19 /media/SSD/Mail/Sieve/thomas.addams_g/.dovecot.sieve
-rw-rw----  vmail  vmail  Jul 14 18:49 /media/SSD/Mail/Sieve/silvia.addams_t/.dovecot.sieve
-rw-rw----  vmail  vmail  Jul 14 16:26 /media/SSD/Mail/Sieve/silvia.addams_g/.dovecot.sieve
-rw-rw----  vmail  vmail  Jul 14 18:49 /media/SSD/Mail/Sieve/manuel.addams_t/.dovecot.sieve
-rw-rw----  vmail  vmail  Jul 14 16:26 /media/SSD/Mail/Sieve/manuel.addams_g/.dovecot.sieve
```

15. Getmail-Eventhandler

In diesem Kapitel wird ein Eventhandler eingerichtet, der immer bei einem bestimmten eintretenden Ereignis seine Arbeit aufnimmt. Was bedeutet das? Das ist relativ einfach zu erklären.... ein Event ist übersetzt ein Ereignis, und zwar ist hier damit das Ereignis einer zeitlich willkürlichen Anmeldung eines Anwenders in seine persönlichen Postfächer gemeint. Betrachten wir dazu einmal die Anmeldung eines Users aus Sicht des Servers. Zunächst mal ist festzustellen, dass der Server nie weiß, ob und wann sich ein User anmeldet, ob sich nur ein einzelner User anmeldet oder nacheinander oder gleichzeitig viele User. Für den Server sind alle über den Tag verteilten Anmeldungen bezogen auf die gerade aktuelle Uhrzeit der Anmeldung rein zufällig eintretende Ereignisse, ein Vorgang, der sich nie exakt zur gleichen Zeit wiederholen wird.

Speziell auf unseren Mailserver geschaut bedeutet "*Anmeldung*", dass ein User einen Mail-Client (z.B. Thunderbird) gestartet hat und dieser Mail-Client verbindet sich im Hintergrund automatisch zum Mail-Server, um einen IMAP-Zugang zu den Mail-Konten dieses Users zu bekommen. Aus Sicht des Servers ist diese Anmeldung ein unvorhergesehenes, ungeplantes, aber erlaubtes Ereignis. Der Server erhält im Anmelde-Prozess vom Client-PC einen Anmeldenamen und ein zugehöriges Password, um damit zu überprüfen, ob dieser User sich überhaupt anmelden darf. Und erst nach diesem Anmelde-Ereignis sollen passend zu den weiter oben beschriebenen Erklärungen nun über den Eventhandler und nur für diesen Anwender die Emails aller seiner ISP-Email-Konten abgerufen werden und in die Dovecot-Postfächer transportiert werden.

Wer sich jedoch zuvor schon entschlossen hat, die Emails aller Konten einfach zyklisch per Cron abzurufen, braucht sich das hier folgende nicht weiter antun und kann dieses Kapitel einfach überspringen. Oder wer seinen Email-Server von außerhalb erreichbar gemacht hat, also über das Internet, sollte auch besser über einen Timer-IRQ oder alternativ auch über eine Lösung mit Cron nachdenken. Bei solchen Rahmenbedingungen würde ich tatsächlich auch nicht weiter über einen vom User auf dem Client ausgelösten Interrupt auf dem Server nachdenken, um darüber dann den Download der persönlichen Emails zu veranlassen.

Wer jedoch die Idee mit dem Eventhandler gut findet und das auch gerne so realisieren möchte… nun ja, der muss sich halt damit auseinandersetzen. Was ist das Ziel? Was ist dafür zu tun? Wie ist der Ablauf? Fangen wir mit dem Ablauf an, um zu sehen, was passiert, wenn ich mich zum Beispiel als Anwender auf meinem PC anmelde, und der am Ende auch gleich das Ziel beschreibt:

Dovecot bietet bei Eintritt eines solchen Anmelde-Ereignisses über ein Post-Login-Script die Möglichkeit an, nach einer erfolgten Anmeldung ein externes Programm zu starten. Der Name "Post Login" sagt es schon aus... es bedeutet "Nach Login" und beschreibt, wann das externe Programm gestartet wird. Das eigentliche Anmelde-Ereignis findet natürlich in Dovecot statt und Dovecot hat für dieses Ereignis eine interne Handlerroutine. Aber praktisch und auch faktisch gesehen ist hier in diesem Fall das gestartete externe Programm ebenfalls ein Event-Handler, weil es ja tatsächlich nur bei Eintritt eines Anmelde-Ereignisses gestartet wird.

Und wenn wir uns jetzt daran erinnern, dass es User mit mehreren Email-Konten gibt, wie z.B. die schon oben genannten Email-Adressen thomas.addams_t@mail und thomas.addams_g@mail, dann bedeutet, dass es sich aus Sicht des Users Thomas Addams zwar beides Male um die gleiche Person handelt, aber gleichzeitig aus Sicht des Servers zwei Anmeldungen passieren, für jedes Konto erfolgt eine eigene Anmeldung. Und es können ja auch durchaus für einen User noch weitere Konten eingerichtet sein. In der weiteren Folge dieses IT-Prozesses bedeutet das, das der Dovecot-Post-Login-Handler für jede Anmeldung das externe Programm startet, also unseren Getmail-Eventhandler, um von diesem Programm die Emails für das jeweilige betroffene Email-Konto beim ISP via POP3 abzurufen.

Im Normalfall ist das eigentlich unkritisch.... es laufen einfach nur mehrere Getmail-Eventhandler-Prozesse gleichzeitig ab, für jede Anmeldung und für jedes lokale Email-Konto ein Prozess. Würden sich gleichzeitig und völlig zufällig auch noch ein zweiter und dritter User anmelden, und jeder dieser User hat auch wieder mehrere Konten, würden alle Prozesse parallel gestartet werden.

Es ist nicht zu verhindern, das das externe Programm Getmail-Eventhandler durch Dovecot für jede Anmeldung an ein Emailkonto parallel gestartet wird. Auch dann nicht, wenn es sich für mehrere Konten

um die gleiche Person handelt. Im Regelfall wird Dovecot ja eigentlich davon ausgehen, dass ein User immer nur ein Email-Konto hat. Wenn ich im Vergleich dazu bei GMX mehrere Konten für mich eingerichtet habe, weiss GMX ja auch nicht, ob bei der nacheinander erfolgenden Anmeldung in diese Konten nur eine Person dahinter steckt oder ob es mehrere sind.

Aber speziell bei diesem Umstand einer möglicherweise parallelen Last habe ich mich damals auch daran erinnert, dass der Mailserver ja nur auf einem Raspberry PI laufen sollte, der ja nicht gerade ein Rechengeschwindigkeitswunder ist. Aus diesem Grund habe ich mich dann entschieden, die Prozesse für einen User (also mit einem oder mehreren Konten) nicht zu parallelisieren, sondern die Konten seriell abzurufen. Mehrere User gleichzeitig sind natürlich parallel, das ist nicht zu verhindern, aber mehrere Konten eines Users werden seriell nacheinander bearbeitet. Insofern ist es auch wieder unkritisch, wenn sich mehrere User gleichzeitig anmelden.

Der technische Ablauf ist also wie folgt:

- für jede Kontoanmeldung wird das externe Programm Getmail-Eventhandler als eigene Instanz gestartet
- die erste Programminstanz ermittelt über die alias-maps im Dovecot-Verzeichnis den zur Dovecot-Anmeldung passenden Linux-User und erstellt ein Lockfile.
- alle weiteren Instanzen ermitteln ebenfalls über die alias-maps "ihren" Linux-User und prüfen dann, ob für diesen Linux-User bereits ein Lockfile besteht. Und wenn ja, beendet sich das Programm umgehend.
- die erste Programminstanz, die einzige, welche die eigentlich Aufgabe *Abholung der Emails* durchführt, ermittelt aus dem Getmail-Verzeichnis des Users alle ISP-Konten, bearbeitet nacheinander alle diese Postfächer und beendet sich nach erfolgreichen Download der Emails.

Der Anmelder startet auf seinem Client-PC innerhalb des Netzwerkes seinen Thunderbird-Client, was quasi das Startsignal für einen automatisch ablaufenden Prozess bedeutet







Die Aktion löst über das Netzwerk die Anmeldung an den Server aus



Die Dovecot-Instanz auf dem Server startet das externe Programm Getmail-Eventhandler für den angemeldeten User .

Der gestartete Job ermittelt den Linux-User für den angegebenen virtuellen User und sucht anschließend im persönlichen Verzeichnis /media/SSD/Mail/Getmail/thomas nach den zuvor erstellten *.conf-Files, um diese einzeln abzuarbeiten und mit den dort eingetragen Anmeldedaten die Emails von den Mail-ISP abzurufen und in thomas‘ persönliche Postfächer unseres Mailserver zu übertragen.



Der Getmail-Eventhandler - zuerst die Datei erstellen und anschließend nicht vergessen, die Rechte zu setzen:

```
# ne /usr/local/bin/getmail_eventhandler
# chown root:root /usr/local/bin/getmail_eventhandler
# chmod 755 /usr/local/bin/getmail_eventhandler
```

Zur Erinnerung: alle notwendigen Dateien können mit diesem Archiv herunter geladen werden:
http://www.thlu.de/Public/TomsMailserver.tar

```
#!/bin/bash
#============================================================================================
# Script-Name :  getmail-eventhandler
# Date        :  25.04.2022
# Version     :  4.3.2
# Lizenz      :  GNU General Public License 3
#
# Description :  Eventhandler, startet by dovecot after a virtual-dovecot-user-login.
#                Than starts downloading the existing Emails from ISP-Mail-Provider
#                for all accounts of this user.
#
# Dependencies : getmail, dovecot (core, imapd, lmtpd), postfix
#============================================================================================

PATH=/usr/local/sbin:/usr/local/bin:/usr/sbin:/usr/bin:/sbin:/bin

VirtUser=""
LinuxUser=""
LockFile=""
GetmailConfDir=""

msg0="process canceled! no suitable linux-user found in alias_maps:"
msg1="processing downloads for:"
msg2="process canceled! no suitable linux-user found in passwd:"
msg3="multiple download emails refused, because already locked for:"
msg4="start download emails at:"
msg5="process canceled! mailbox-conf-files not found."
msg6="append lock-file-entry for:"
msg7="remove lock-file for:"

[ -z "$(which getmail)" ] && echo "Error: getmail not found! Job terminated." | systemd-cat -t "thlu:$(basename $0)" -p "info" && exit 1

#============================================================================================

DoPollingByAccount()
{
    anyconffound=false

    echo "$msg1 $LinuxUser" | systemd-cat -t "thlu:$(basename $0)" -p "info"
    echo "using getmail-conf-directory: $GetmailConfDir" | systemd-cat -t "thlu:$(basename $0)" -p "info"

    if [ -d "$GetmailConfDir" ]; then
        if [ -n "$(find $GetmailConfDir/*.conf -maxdepth 0 -print 2> /dev/null)" ]; then
            anyconffound=true
        fi
    fi

    if [ ! -f $GetmailConfDir/getmail-last-polling.log ]; then
        touch $GetmailConfDir/getmail-last-polling.log
        chown vmail:vmail $GetmailConfDir/getmail-last-polling.log
    fi
```

```
    if [ $anyconffound == true ]; then
        echo "$msg4 $(date +%d.%m.%Y) - $(date +%H:%M:%S)" >$GetmailConfDir/getmail-last-polling.log
        while read line
        do
            echo "processing getmail: started downloads for $line, setuid to user vmail (UID 5000) " | systemd-cat -t "thlu:$(basename $0)" -p "info"
            su vmail -m -p -c "getmail -v --getmaildir $GetmailConfDir --rcfile $line 2>&1 >>$GetmailConfDir/getmail-last-polling.log"
        done < <(ls $GetmailConfDir/*.conf)

        sleep 60
        echo "$msg7 $LinuxUser" | systemd-cat -t "thlu:$(basename $0)" -p "info"
        rm "$LockFile"
    else
        echo "$msg5" | systemd-cat -t "thlu:$(basename $0)" -p "warning"
    fi

    return 0
}
#=============================================================================================
# Main

VirtUser=$(echo "$USER" | sed 's/@mail//; s/^[ \t]*//; s/[ \t]*$//')
LinuxUser=$(awk '$1 == "'$VirtUser'" {print $2;exit}' /etc/dovecot/alias_maps)

if [ -z "$LinuxUser" ]; then
    echo "$msg0   Virtuser=$VirtUser" | systemd-cat -t "thlu:$(basename $0)" -p "err"
else
    LinuxUser_UID=$(/usr/bin/id -r -u "$LinuxUser" 2>/dev/null)

    if [ -z "$LinuxUser_UID" ]; then
        echo "$msg2  VirtUser=$VirtUser  LinuxUser=$LinuxUser" | systemd-cat -t "thlu:$(basename $0)" -p "err"
    else
        echo "started: VirtUser=$VirtUser  LinuxUser=$LinuxUser  UID=$LinuxUser_UID" | systemd-cat -t "thlu:$(basename $0)" -p "info"

        GetmailConfDir="/media/SSD/Mail/Getmail/$LinuxUser"
        LockFile="$GetmailConfDir/getmail.lock"
        find "$LockFile" -mmin +60 -exec rm "$LockFile" \; 2>/dev/null

        echo "$msg6 $VirtUser ($LinuxUser)" | systemd-cat -t "thlu:$(basename $0)" -p "info"
        echo "$VirtUser.$$" >>$LockFile
        line=$(head -n 1 $LockFile)

        if [ "$line" == "$VirtUser.$$" ]; then
            DoPollingByAccount &
        else
            echo "$msg3 $LinuxUser ($VirtUser)" | systemd-cat -t "thlu:$(basename $0)" -p "warning"
        fi
    fi
fi

exec "$@"
exit 0

#=============================================================================================
# Ende
```

16. Abschlussarbeiten

Fertig! Es ist vollbracht. Alle für den Mailserver notwendigen Arbeiten sind meines Erachtens erfolgt. Und ich gestehe, dass es mir nun ein wenig mulmig wird, wenn ich auf die Seitenzahl direkt oberhalb schaue. Es ist deutlich umfangreicher geworden als zu Anfang gedacht und von mir geplant. Aber wenn ich den Anspruch habe, die Zusammenhänge auch wirklich zu verstehen und dann mit diesem Wissen willentlich, bewusst, zielgerichtet und mit Sachverstand meinen Server betreiben will, dann weiß ich nicht, welche Erklärung hier so unwichtig ist, dass man sie einfach streichen kann.

Egal, es ist so, wie es ist, und eben manchmal auch übers technische hinaus ein wenig langatmig. Mein Anspruch war, nicht nur das Wie zu erläutern, sondern vor allem auch das Warum. Das, was ich tue, soll einen plausiblen Grund haben und sinnvoll sein. Und all das mit dem Gedanken, wenn ich zu wenig Wissen habe, steigt die Gefahr mit dem Projekt zu scheitern. Dem entgegen kann man nie zu viel Wissen haben - Wissen verbessert sogar ganz sicher die Aussicht auf Erfolg. Aber nun ist der Moment der Wahrheit gekommen, weil jetzt diese Anleitung beweisen muss, ob ein Erfolg damit überhaupt möglich ist und ob sie was taugt - oder ob sie vielleicht doch besser nur taucht.

Wir melden uns auf dem Server via ssh an und führen nun eine kleine Diagnostik durch und checken dabei in einer Funktionsprüfung alle beteiligten Komponenten. Wir beginnen mit der Kontrolle, ob die Postfix-Rules aktuell sind und der jeweiligen Quelldatei entsprechen:

```
root@raspi3:/etc/postfix
# ls /etc/postfix/rules/*
-rw------- 1 root root 121 2017-06-13 18:13 /etc/postfix/rules/check_generic_replace_from
-rw------- 1 root root 999 2017-06-13 18:13 /etc/postfix/rules/get_relayhost_passwd
-rw------- 1 root root 918 2017-06-13 18:14 /etc/postfix/rules/check_sender_filter
-rw------- 1 root root 919 2017-06-13 18:14 /etc/postfix/rules/get_sender_relayhost
-rw------- 1 root root 275 2017-07-01 12:29 /etc/postfix/rules/check_local_transport_maps
-rw------- 1 root root 516 2017-06-13 14:32 /etc/postfix/rules/check_local_alias_maps
```

```
# ls /etc/postfix/*.db
-rw-r----- 1 root postfix 12K 2017-07-01 12:30 /etc/postfix/check_generic_replace_from.db
-rw-r----- 1 root postfix 12K 2017-07-01 12:30 /etc/postfix/get_relayhost_passwd.db
-rw-r----- 1 root postfix 12K 2017-07-01 12:30 /etc/postfix/check_sender_filter.db
-rw-r----- 1 root postfix 12K 2017-07-01 12:30 /etc/postfix/get_sender_relayhost.db
-rw-r----- 1 root postfix 12K 2017-07-01 12:30 /etc/postfix/check_local_transport_maps.db
-rw-r----- 1 root postfix 12K 2017-07-01 12:30 /etc/postfix/check_local_alias_maps.db
```

Der Timestamp der *.db-Dateien ist durch die Erstellung mit dem Helperscript refresh-rules bei allen gleich. Wichtig ist aber, dass in der Liste oberhalb keine der Quelldateien jünger ist. In dem Fall wurde oben nachträglich etwas geändert und deshalb ist es erforderlich, das Helperscript erneut zu starten.

Fangen wir mit Postfix an. Der Befehl

```
# postfix check
```

muss ohne Ausgabe beendet werden. Eine Ausgabe bedeutet, dass Fehler vorliegen. Wenn Fehler berichtet werden, geht‘s erst dann weiter, wenn die Fehler behoben sind. Wenn keine Fehler berichtet werden, starten wir den Dienst und prüfen erneut auf Fehler über den Status.

Nun werden die beiden laufenden Dienste gestoppt, da wir jetzt gar nicht mehr zuverlässig wissen, ob in den letzten Start der Dienste wirklich schon alle Änderungen eingeflossen sind.

```
# systemctl stop postfix dovecot
```

Die beiden Services werden nun einzeln nacheinander gestartet. Auch die folgenden Ausgaben dürfen keine Fehler enthalten. Fehler sind eigentlich mit etwas Sorgfalt beim Ansehen immer auch sofort als Fehler zu erkennen. Die relevanten Meldungen ohne Fehler sehen gekürzt etwa so aus:

systemctl start postfix; systemctl status postfix

```
Jul 02 18:36:46 raspi3 postfix[19809]: Starting Postfix Mail Transport Agent: postfix.
Jul 02 18:36:46 raspi3 systemd[1]: Started LSB: Postfix Mail Transport Agent.
Jul 02 18:36:46 raspi3 postfix/master[19922]: daemon started -- version 2.11.3,
```

Als nächstes wird Dovecot gestartet und ebenfalls kontrolliert:

systemctl start dovecot; systemctl status dovecot

```
Jul 02 18:42:00 raspi3 dovecot[19985]: master: Dovecot v2.2.13 starting up for imap, lmtp (core dumps disabled)
Jul 02 18:42:00 raspi3 dovecot[19979]: Starting IMAP/POP3 mail server: dovecot.
Jul 02 18:42:00 raspi3 systemd[1]: Started LSB: Dovecot init script.
```

Beide Prozesse sind erfolgreich und ohne Fehler gestartet und sind nun aktiv. Das wars! Als zunächst letzte Maßnahme fehlt jetzt nur noch ein einmaliger Neustart des Systems und anschließend ein letzter Blick auf den Status der Dienste.

systemctl reboot

Und nach dem Reboot wieder die Kontrolle, die beiden Services müssen Fehlermeldungen laufen:

systemctl status dovecot postfix

Danach werfen wir noch mal einen kurzen Blick auf die Fehlersituation des Systemstarts an sich:

root@raspi3:~
journalctl -b -p err

Wenn hier andere Fehler berichtet werden, so würde ich das unbedingt als Anlass nehmen, nach Ursache und Bedeutung zu suchen. Und wenn es möglich ist, sollten die Fehler auf jeden Fall beseitigt werden. Sie einfach zu ignorieren ist die denkbar schlechteste Entscheidung.

17. Thunderbird

Jetzt fehlt nur noch die Einrichtung der neuen Postfächer unseres Mailservers in Thunderbird. Aber das ist schnell gemacht. Die für die Anwender eingerichteten Postfachnamen kenne ich alle… ich hab sie ja selber eingerichet. Und die Postfach-Passwörter (wir erinnern uns, siehe Seite 27-29) entsprechen hier in diesem Beispiel dem Linux-Password des Anwender plus der Erweiterung „Mailsys-rpi3“.

Mein Username und das Password auf dem Server (für Linux und Samba) ist das gleiche, wie auf meinem Client-PC (egal obs ein Linux oder Windows-PC ist). Wenn *toms_passwd* also das Linux-Passwort ist, so lautet mein Mailserver-Password: *Mailsys-rpi3.toms_passwd*. Ich beziehe mich natürlich jetzt hier nur auf meine Bespiele und folge konsequent dem zu Anfang angekündigten roten Faden. Wenn die Regeln für die Passworte geändert wurden, kein Problem, es muss halt hier nur der korrekte Username und das korrekte Password der Dovecot-DB eingetragen werden. Aber beides sollte ja bekannt sein.

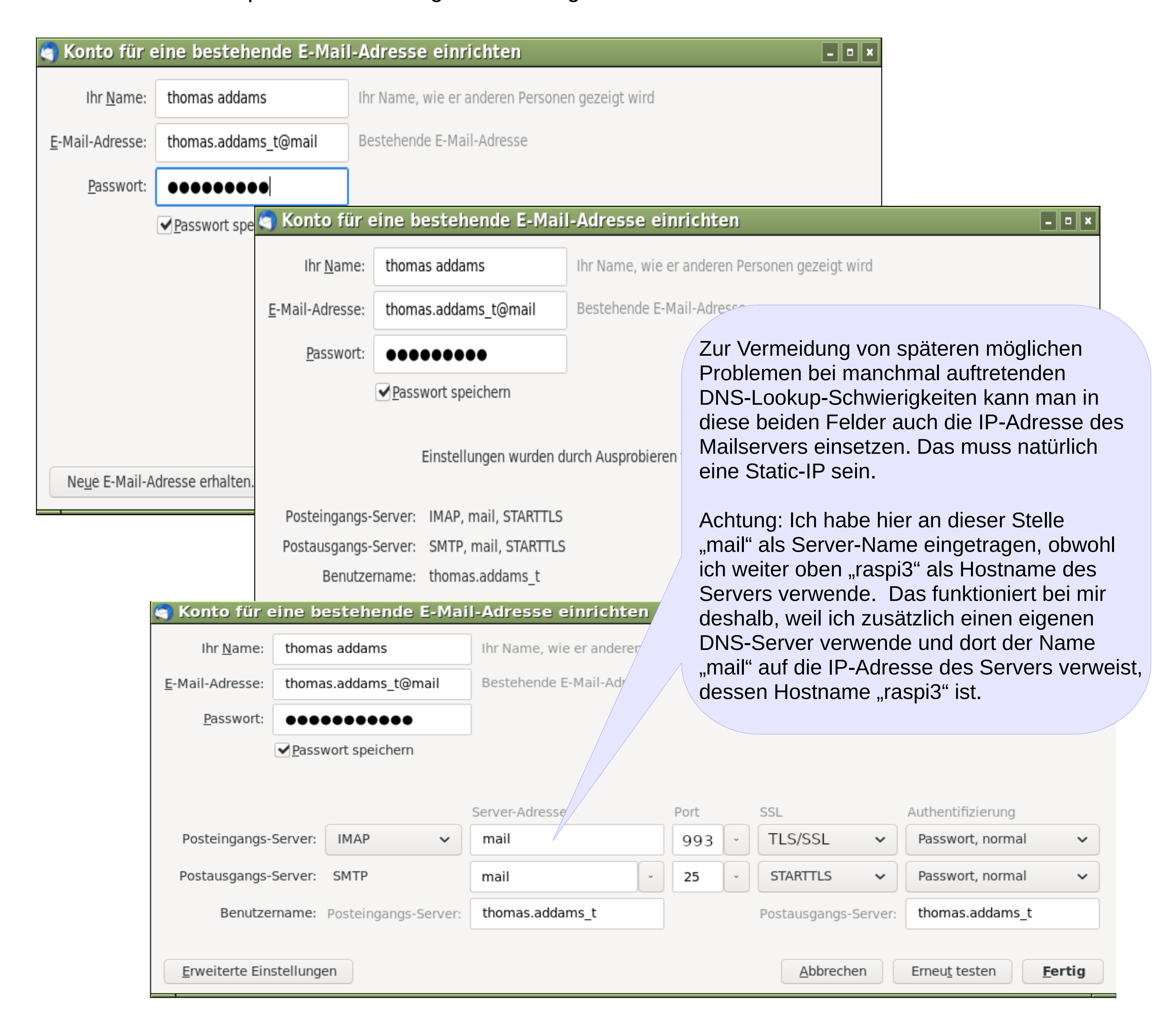
Jetzt wähle ich Thunderbird einfach über das Start-Menü oder über einen Desktop-Starter aus. Und nachdem Thunderbird bereit ist, wähle ich über

Einstellungen > Konteneinstellungen > Konten-Aktionen

die Option

Email-Konto hinzufügen

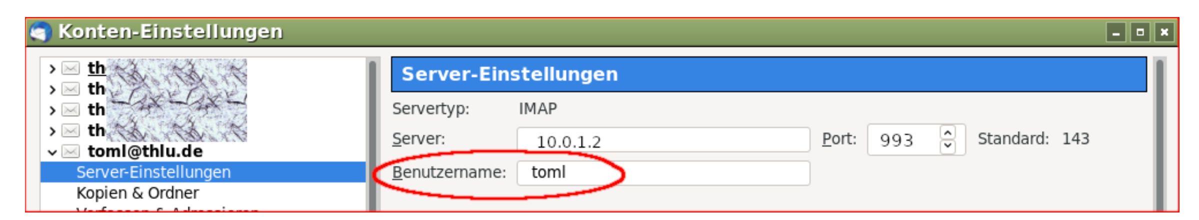
aus und erstelle die passenden Einträge in den Eingabefeldern:



Der Benutzername im Thunderbird-Konto muss identisch mit dem sein, der in der Dovecot->passwd eingetragen ist. Falls die Anmeldung fehlschlägt oder im Journal des Mailserver die folgende Meldung auftaucht...

```
# journalctl -b -p err | grep getmail
Aug 19 17:23:39 raspi3 thlu:getmail_eventhandler[4745]: process canceled! no suitable linux-user found in alias_maps: Virtuser=thomas.addams_t@mail
```

...kann hier auch ein falscher Anmeldename verwendet worden sein. Der Anmeldename lautet nicht thomas.addams_t@mail, sondern thomas.addams_t. Das kann nachträglich in den Konto-Einstellungen geändert werden, hier als Beispiel für mein Konto „toml“:



Unmittelbar bei oder nach der Einrichtung des Thunderbird-Kontos wird sofort auch unser Server-Zertifikat abgefragt, welches wir bei der Einrichtung von Dovecot erstellt haben. Wir können uns hier mit „Ansehen“ den Inhalt anzeigen lassen und sehen quasi die Einträge, die wir zuvor bei der Erstellung des Zertifikats vorgegeben haben.

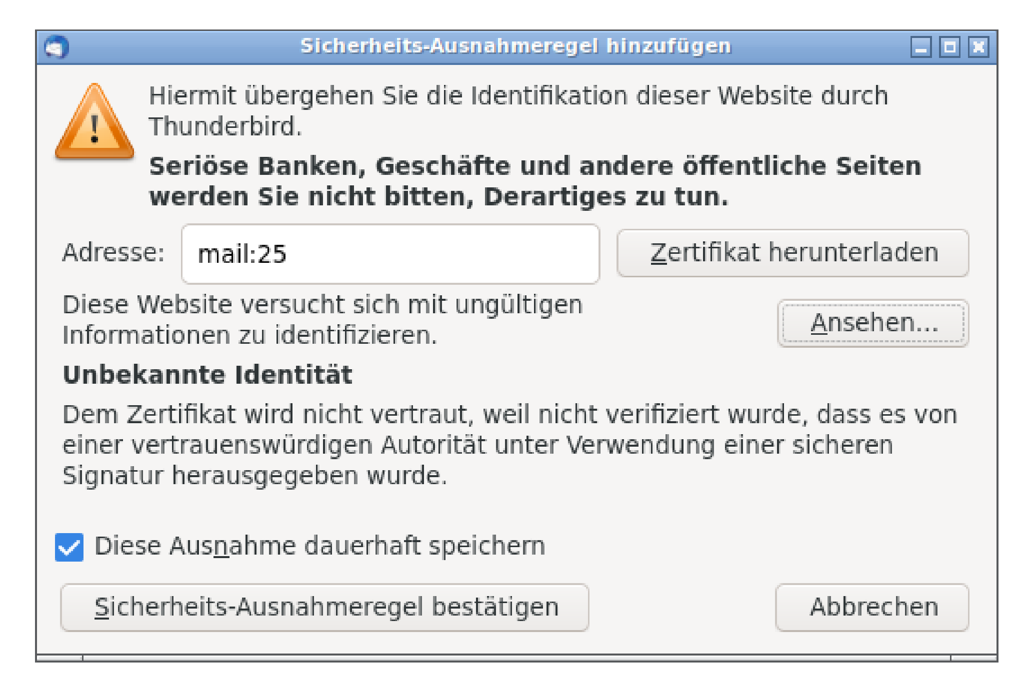
Das es sich um eigene private Zertifikate handelt, ohne die für große Unternehmen obligatorische Glaubwürdigkeits- bzw. Vertrauenszertifizierung können wir hier nur die Sicherheits-Ausnahmeregel bestätigen.



Achtung, hier noch ein wichtiger Hinweis. Der allererste Versuch eine Email mit dem neuen Postfach zu senden kann eventuell auch fehlschlagen.

Thunderbird wird den ersten Sendevorgang vielleicht mit einer Fehlermeldung abbrechen. Das ist nichts tragisches, sondern sogar vorhersehbar. Thunderbird bricht ab, weil keine verschlüsselte Verbindung zu unserem SMTP-Server aufgebaut werden kann. Das klappt deshalb nicht, weil es auf einem im Hintergrund geöffneten Fenster zunächst auch für den Sendevorgang über Postfix das Zertifikat einmalig bestätigt haben möchte.

Obwohl beide Dienste das gleiche Zertifikat und Key verwenden, hatte ich auf der Testmaschine in der Entwicklung zeitweise die doppelte Abfrage. Wenn der Sendevorgang also tatsächlich aus diesem Grund fehlschlägt, einfach „abbrechen“ anwählen, im anderen Fenster wie zuvor für Dovecot auch für Postfix das Zertifikat bestätigen und die Mail erneut senden. Danach und auch für alle weiteren Sendevorgänge wird die Mail fehlerlos versendet werden.



18. Funktionsprüfung

Nun prüfen wir, ob Abfragen und Transporte funktionieren. Wir benötigen dazu ein Terminal-Fenster, für den SSH-Zugang zum Server und starten dort die Journal-Anzeige im Follow-Mode:

root@raspi3:~
journalctl -f

```
-- Logs begin at Do 2017-06-15 10:51:34 CEST. --
Jan 12 11:32:41 raspi3 systemd[798]: Startup finished in 605ms.
Jan 12 11:32:41 raspi3 systemd[1]: Started User Manager for UID 1000.
Jan 12 11:32:45 raspi3 su[820]: Successful su for root by thomas
Jan 12 11:32:45 raspi3 su[820]: + /dev/pts/0 thomas:root
Jan 12 11:32:45 raspi3 su[820]: pam_unix(su:session): session opened for user root by thomas(uid=1000)
Jan 12 11:32:49 raspi3 systemd[1]: Starting Dovecot IMAP/POP3 email server...
Jan 12 11:32:49 raspi3 systemd[1]: Started Dovecot IMAP/POP3 email server.
Jan 12 11:32:50 raspi3 systemd[1]: Starting Postfix Mail Transport Agent...
Jan 12 11:32:50 raspi3 systemd[1]: Started Postfix Mail Transport Agent.
```

Jetzt initiieren wir von unserem PC aus einen Abruf-Vorgang, in dem wir Thunderbird starten Das abzurufende Postfach beim Mail-ISP ist derzeit leer, dessen habe ich mich jetzt unmittelbar zuvor über den Browser und einer kurzen Anmeldung im Postfach auf der ISP-Web-Site vergewissert. Wir starten Thunderbird und beobachten gleichzeitig das SSH-Terminalfenster auf Fehler.

```
Jan 12 11:32:49 raspi3 systemd[1]: Starting Dovecot IMAP/POP3 email server...
Jan 12 11:32:49 raspi3 systemd[1]: Started Dovecot IMAP/POP3 email server.
Jan 12 11:32:50 raspi3 systemd[1]: Starting Postfix Mail Transport Agent...
Jan 12 11:32:50 raspi3 systemd[1]: Started Postfix Mail Transport Agent.

Jan 12 11:34:02 raspi3 dovecot[451]: imap-login: Login: user=<thomas.addams_t>, method=PLAIN, rip=10.0.1.58, lip=10.0.1.58, mpid=730, TLS
Jan 12 11:34:02 raspi3 thlu:getmail_eventhandler[740]: append lock-file-entry for: thomas.addams_t (thomas)
Jan 12 11:34:02 raspi3 thlu:getmail_eventhandler[745]: processing downloads for: thomas
Jan 12 11:34:02 raspi3 thlu:getmail_eventhandler[748]: using getmail-conf-directory: /media/SSD/Mail/Getmail/thomas
Jan 12 11:34:02 raspi3 thlu:getmail_eventhandler[759]: processing getmail: /media/SSD/Mail/Getmail/thomas/mailbox01.conf
Jan 12 11:34:02 raspi3 thlu:getmail_eventhandler[768]: started downloads for mailbox01.conf by root, setuid to user thomas ( 1000)
Jan 12 11:34:02 raspi3 thlu:getmail_eventhandler[777]: processing getmail: /media/SSD/Mail/Getmail/thomas/mailbox02.conf
Jan 12 11:34:02 raspi3 thlu:getmail_eventhandler[784]: started downloads for mailbox02.conf by root, setuid to user thomas ( 1000)

Jan 12 11:34:02 raspi3 dovecot[451]: imap-login: Login: user=<thomas.addams_g>, method=PLAIN, rip=10.0.1.58, lip=10.0.1.58, mpid=750, TLS
Jan 12 11:34:02 raspi3 thlu:getmail_eventhandler[773]: append lock-file-entry for: thomas.addams_g (thomas)
Jan 12 11:34:02 raspi3 thlu:getmail_eventhandler[781]: download emails refused, because already locked for: thomas (thomas.addams_g)

Jan 12 11:38:02 raspi3 thlu:getmail_eventhandler[910]: remove lock-file for: thomas

Jan 12 11:42:21 raspi3 dovecot[451]: imap(thomas.addams_g): Logged out in=158 out=686
Jan 12 11:42:21 raspi3 dovecot[451]: imap(thomas.addams_t): Logged out in=209 out=788
```

Was sagen uns diese Log-Einträge? Das ist einfach erklärt... es haben sich zwei User angemeldet, oder besser gesagt, ich habe mich mit zwei Accounts angemeldet. Der Account, der zufällig von Dovecot als erstes angemeldet wird, startet den Download-Prozess für alle Konten des ermittelten Linux-Users, hier "thomas". Bei der Anmeldung des zweiten Accounts wird festgestellt, dass für den Linux-User "thomas" bereits ein Job für alle Konten läuft, also werden weitere Instanzen abgebrochen.

Nun führe ich den gleichen Test noch einmal durch. Allerdings habe ich jetzt zuvor eine Email an das abzurufende Postfach thomas.addams@toml.de gesendet. Und was passiert im Journal…?… das gleiche, wie zuvor – es läuft ohne Fehler durch, und wir sehen zusätzlich eine neue Nachricht:

```
Jan 12 14:09:40 raspi3 dovecot: lda(thomas.addams_t): sieve: msgid=<ade7d773@mail>: stored mail into mailbox 'INBOX'
```

Die Mail ist angekommen und wir finden sie in der Inbox:

ls /media/SSD/Mail/Imap/thomas/thomas.addams_t/new

```
insgesamt 20K
drwx------ 2 thomas thomas 4,0K 2018-01-12 14:09 .
drwx------ 5 thomas thomas 4,0K 2018-01-12 14:09 ..
-rw------- 1 thomas thomas 1,2K 2018-01-12 14:09 1499161377.1376_0.mail
```

In der Inbox ist eine Nachricht gespeichert und ich prüfe, ob es wirklich meine Test-Nachricht ist:

egrep -i "test-nachricht|from:" 1499161377.1376_0.mail

```
From: "thomas@mailisp.de" <thomas@mailisp.de>
Test-Nachricht: 12.01.18 – 14:00
```

„Test-Nachricht: 12.01.18 – 14:00“ war der Inhalt der Mail im Textfeld. Mit anderen Worten, meine Test-Mail wurde perfekt und fehlerlos zugestellt.

Wer weiteres Interesse daran hat, noch ein besseres Gefühl für die Abläufe und die Zusammenhänge zu bekommen, kann nun noch einen weiteren Test durchführen. Und zwar den manuellen Transport allein auf der Server-Seite.

Dazu senden wir noch einmal wie zuvor eine Mail an das Postfach und wechseln anschließend in das SSH-Terminalfenster. Der Abruf wird natürlich mit der UID des Users „vmail“ durchgeführt:

root@raspi3:~ (als root, zusammenhängend als 1 Zeile)

```
# su vmail -m -p -c "getmail -v --getmaildir /media/SSD/Mail/Getmail/thomas
--rcfile /media/SSD/Mail/Getmail/thomas/mailbox01.conf"
```

```
getmail version *
Copyright (C) 1998-2012 Charles Cazabon.  Licensed under the GNU GPL version 2.
SimplePOP3SSLRetriever:thomas.addams@pop3.toml.de:995:
  msg 1/1 (1047 bytes) delivered, deleted
  1 messages (1047 bytes) retrieved, 0 skipped
```

Auch diese Mail liegt nun in der Inbox des Postfaches. Wir können jetzt ganz am Ende angekommen und der Vollständigkeit wegen auch noch mal einen Blick in die Transport-Logs werden. Gerade die Kenntnis vom Vorhandensein dieser Logs ist enorm wichtig. Möglicherweise entsteht irgendwann mal eine Fehlersituation und dann braucht man alle Hilfen, die es gibt, um die Fehlerursache festzustellen. Die Logs sind bei der Fehlersuche immer unbedingt eine Hilfe.

root@raspi3:~
```
# ls /media/SSD/Mail/Getmail/thomas
```

```
insgesamt 20K
drwx------ 2 thomas thomas 4,0K 2018-01-12 14:09 .
drwx------ 4 thomas thomas 4,0K 2018-01-12 14:09 ..
-rw-r--r-- 1 thomas thomas  304 2018-01-12 14:09 getmail-last-polling.log
-rw-r--r-- 1 thomas thomas  735 2018-01-12 14:09 thomas_addams_at_toml_de.log
```

cat /media/SSD/Mail/Getmail/thomas/getmail-last-polling.log

```
start download emails at: 12.01.2018 - 14:09
getmail version *
Copyright (C) 1998-2012 Charles Cazabon.  Licensed under the GNU GPL version 2.
SimplePOP3SSLRetriever:thomas.addams@pop3.toml.de:995:
  msg 1/1 (1047 bytes) delivered, deleted
  1 messages (1047 bytes) retrieved, 0 skipped
```

cat /media/SSD/Mail/Getmail/thomas/thomas_addams_at_toml_de.log

```
2018-01-12 14:09:34 msg 1/1 (2538 bytes) msgid 1496851323.7 from <thomas.addams@pop3.toml.de>
delivered to MDA_external command deliver (), deleted
```



Das wars… fertig… alles geschafft.

Da ja jetzt alle Dienste anscheinend ohne Fehler laufen und auch im Journal nichts außergewöhnliches berichtet wird, können wir einmal tief durchatmen und uns entspannt zurücklehnen. Das Ziel ist erreicht, der eigene Mailserver ist vollständig eingerichtet und betriebsbereit.



19. Der Server

Mein Mail-Server lief über Jahre absolut zufriedenstellend auf einem Raspberry Pi, Pi 2 Model B, mit der ARM Cortex-A7 CPU. Als Betriebssystem war natürlich Raspian Stretch installiert – und zwar OHNE Desktop-Environment. Auf dem aktuellen System, ein Intel NUC (Mini-PC) gelten heute unverändert die gleichen Prämissen: kein grafischer Desktop, keine unnötigen Dienste. Aktuell ist allerdings Debian „Stable“ das laufende Betriebssystem.

Und genau diese Empfehlung möchte ich hier am Ende noch aussprechen. Bei solchen Projekten halte ich es grundsätzlich für notwendig, die „Oberfläche“ einer solchen Maschine für mögliche Angriff so klein wir nur irgendwie möglich zu halten. Das bedeutet, auf einem als Server wirkenden Linux-PC ist ein grafisches Desktop-Environment nicht nur unnötig, es hat meiner Meinung nach dort überhaupt nichts zu suchen.

Ich empfehle hier die Installation ohne grafischen Desktop und darüber hinaus, zusätzlich alle unbenötigten Dienste rigoros zu entfernen. Danach werden nur die für die gewünschten Aufgaben benötigten Programme installiert. Die Liste der aktiven Prozesse unmittelbar nach der Installation ist bei mir ziemlich kurz:

```
LXTerminal
Datei  Bearbeiten  Reiter  Hilfe

  1  [                                        0.0%]   Tasks: 19, 5 thr; 1 running
  2  [||                                      2.0%]   Load average: 0.03 0.03 0.00
  3  [                                        0.0%]   Uptime: 15 days, 02:58:40
  4  [                                        0.0%]
  Mem[||||||||||||||||                  141M/7.39G]
  Swp[|                                8.75M/9.31G]

  PID USER      PRI  NI  VIRT   RES   SHR S CPU% MEM%   TIME+  Command
25214 root       20   0  8020  3728  3048 R  0.7  0.0  0:00.31 htop
    1 root       20   0  166M  5748  3436 S  0.0  0.1  0:36.79 /sbin/init
  274 root       20   0 67308  4584  3548 S  0.0  0.1  0:16.32 /lib/systemd/systemd-journald
  297 root       20   0 22540  1580  1188 S  0.0  0.0  0:03.72 /lib/systemd/systemd-udevd
  472 systemd-t  20   0 92860  1176   372 S  0.0  0.0  0:04.56 /lib/systemd/systemd-timesyncd
  479 root       20   0 19520  2820  2016 S  0.0  0.0  0:06.20 /lib/systemd/systemd-logind
  483 messagebu  20   0  9316  1536   760 S  0.0  0.0  1:38.64 /usr/bin/dbus-daemon --system --address=systemd: --nofork --n
  485 root       20   0  8800  1120   960 S  0.0  0.0  0:32.25 /usr/sbin/cron -f
  493 root       20   0  5612   432   320 S  0.0  0.0  0:00.00 /sbin/agetty -o -p -- \u --noclear tty1 linux
  524 colord     20   0  243M  3796  2764 S  0.0  0.0  0:00.26 /usr/lib/colord/colord
  894 root       20   0  234M  3388  1912 S  0.0  0.0  0:47.66 /usr/lib/policykit-1/polkitd --no-debug

F1Help  F2Setup F3SearchF4FilterF5Tree  F6SortByF7Nice -F8Nice +F9Kill  F10Quit
```

Nachdem ich das aktuelle Betriebssystem installiert habe, wird die sowieso schon schmale Installation nochmal um einige unnötige Pakete zusätzlich erleichtert:

```
apt purge sudo rsyslog dhcpcd5 networkmanager avahi
apt purge wpasupplicant iw ntp
apt purge pi-bluetooth bluez triggerhappy
```

Hierbei geht es ganz sicher nicht um *„small, smaller, am smallsten gewinnt“* oder um Speicherplatz auf der SDC zu sparen. Das ist natürlich totaler Quatsch. Hier geht es allein darum, die angreifbare Oberfläche des Servers so klein wie möglich zu halten. Hier geht es darum, keine unnötigen Dienste laufen zu lassen. Und mir geht es auch darum, die Bewegungen bei der Erfordernis notwendiger Upgrades auf ein Minimum zu reduzieren – gemäß der Devise „*never touch a running system*“.

Auf meinem Server gibt es ganz sicher keine Bluetooth-Aktivitäten. Weg damit! Wofür dieser Unsinn mit DHCPD, wenn doch der Server sowieso immer läuft und er eh immer die gleiche IPs innehat? Also lieber eine passende Einstellung mit systemd-networkd oder ein Static-IP-Setup vornehmen. Eine doppelte Protokollierung mit rsyslog, wo doch mit dem Journal sowieso schon das perfekte Log läuft - wenn man‘s richtig eingestellt hat? Zwei Programme zur Timersynchronisierung? Wozu das alles? Nein, weg damit!

Sudo ist für mich sowieso ein NoGo, ich bewerte sudo mit seinem per Default eingestellten Password-Keep fast als Exploit. Also hat sudo keinesfalls einen Platz auf meinem Server. WLAN braucht ein solcher Server meiner Meinung nach auch nicht. Am allerwenigsten braucht er eine (vielleicht auch nur vage Möglichkeit), dass er darüber von außen und ohne meine Zustimmung kontaktiert und kompromittiert werden kann.

Und gerade vor dem Hintergrund dieses Sicherheitsgedanken, also der Frage, wie ich meinen Server bestmöglich absichere, habe ich mich über alles andere hinausgehend auch noch dazu entschlossen, den Programmen auf meinem Server KEINEN ausgehenden Zugriff auf das Internet zu erlauben. Den eingehenden Traffic regelt und verhindert seit jeher der Router, den ausgehenden Traffic habe ich seit einiger Zeit vollständig und rigoros verboten. Natürlich ist es ihm technisch grundsätzlich möglich, über meinen Router ganz normal ins Internet zu interagieren. Aber hier vertraue ich einfach auf Debian als Quelle für Raspian und natürlich dem Linux-Kernel, dass meine Einschränkungen respektiert werden.

Meinem Server ist es also nach bestem Wissen und Gewissen und entsprechend meiner Möglichkeiten via Iptables rigoros verboten, Pakete vom Internet zu empfangen oder ins Internet zu senden. Es sind lediglich einzelne explizite Ports erlaubt, z.B. für den Mailserver oder den OpenVPN-Zugang. Alles andere an ein- und ausgehenden TCP-Paketen wird gnadenlos gedropt. Ich vertraue hier natürlich auf die Integrität des Linux-Kernels und das er meine Einstellungen auch wirklich respektiert. Tut er das nicht, gäbe es faktisch überhaupt keine Sicherheit mehr. Dann kann ich es allerdings auch nicht ändern und alle durchgeführten Maßnahmen wären nur noch Blendwerk und Placebos. Aber das glaube ich nicht. Stattdessen glaube ich, das bestmögliche getan zu haben, um meinen Server und unsere Daten zu sichern.

Mein Client-PC

Alle zur Entwicklung dieses Projektes und zur Inbetriebnahme des Mailservers notwendigen Arbeiten, sowie die Erstellung dieser Dokumentation habe ich auf (m)einem Debian-PC durchgeführt. Windows spielt glücklicherweise bei uns überhaupt keine Rolle mehr und ich habe für Windows auch keine Energie verschwendet. Ob und wie diese Arbeiten an einem Linux-Server von einem Windows-PC aus möglich sind…. tja, darüber will ich nicht mal nachdenken… sorry...

Aber wie schon zuvor an anderer Stelle festgestellt, das hier sind eh alles nur meine Anforderungen, meine Überlegungen, meine Gedanken, meine Ideen, meine Entscheidungen, meine Lösungen. Andere Anforderungen erfordern andere Entscheidungen, und natürlich auch andere Lösungen. Und ganz am Ende ist immer die Lösung die richtige, bei der am Ende das rauskommt, was man erwartet und vorgesehen hat.

20. Zu guter Letzt...

… würde ich mich sehr darüber freuen, wenn jemand beim Arbeiten mit dieser Projektbeschreibung über möglicherweise vorhandene sachliche Fehler stolpert oder sie findet und mir dann darüber einen kurzen Hinweis an meine Email-Adresse sendet.

Ich bin natürlich sehr interessiert daran, dass das, was ich hier geschrieben und beschrieben habe, auch im Wesentlichen richtig ist. Und wenn dumme Fehler nachher noch den Erfolg behindern oder gar unmöglich machen, wäre es klasse, wenn ich das korrigieren darf. Ich finde im Moment jedenfalls keine Fehler mehr …. aber das bedeutet nicht, dass es keine mehr gibt, sondern nur, dass ich jetzt an dem Punkt angekommen bin, wo ich vor lauter Bäumen keinen Wald mehr sehe….

Ansonsten bleibt mir nur noch zusagen….

… viel Erfolg!

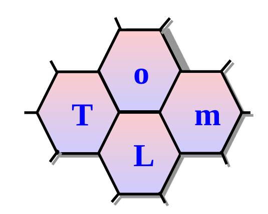
Haftungsausschluss:

Ich erhebe keinen Anspruch darauf, dass diese Dokumentation oder die von mir geschriebenen und im Tar-File enthaltenden Shell-Scripte vollständig fehlerfrei sind und ich behaupte das auch nicht. Sowohl durch Programmierfehler in den Programmen ist Datenverlust möglich, wie auch durch den Anwender selber verursacht durch unsachgemäße oder fehlerhafte Einstellungen oder falsche Bedienung aller hier beschriebenen und verwendeten Programme. Deshalb schließe ich jede Haftung für Schäden an Software oder Hardware oder Vermögensschäden oder für Datenverlust aus, die durch die Benutzung der Shell-Scripte oder dieses Dokumentation entstehen. Die Benutzung dieser Dokumentation sowie aller im Tar-File enthaltenen Dateien erfolgt auf eigenes Risiko.

Dieses Dokument sowie die im Tar-Archiv enthaltenen Dateien enstprechend den Beispielen in diesem Dokument wird veröffentlicht unter:

GNU General Public License 3
im Juli 2017

Aktualisiert: November 2017
Aktualisiert: Januar 2018
Aktualisiert: Oktober 2019
Aktualisiert: August 2022



toml@thlu.de

Epilog

Zertifikate:

Abweichend von einem etablierten und anscheinend auch empfohlenen Standard bei der Organisation von Zertifikaten und deren Benennung nach z.B. folgendem Muster:

/etc/ssl/example.com.key

habe ich mich hier trotzdem für einen eigenen Weg entschieden. Laut Erklärung im Dovecot-Wiki ist es nicht grundsätzlich falsch, eigene Festlegungen zu treffen. Siehe dazu den folgenden Wortlaut:
„By default the certificate is created to /etc/ssl/certs/dovecot.pem and the private key file is created to /etc/ssl/private/dovecot.pem. If you wish to change any of these, modify the mkcert.sh script.“

Das gleiche gilt für Postfix. Auch bei Postfix ist über explizit angegebene Speicherorte für Cert und Key in der main.cf eine Abweichung nach eigenem Ermessen möglich. Postfix verwendet hier in meinem Konzept Zertifikat und Key aus dem Dovecot-Verzeichnis.

Der Hintergrund für meine Entscheidung ist einfach. Ich möchte Änderungen im allgemeinen Betriebssystem nach Möglichkeit nur an den Stellen durchführen, die explizit durch mein Wirken entstanden sind. Hier in diesem Fall also durch die von mir durchgeführte Installation der Pakete dovecot und postfix und der in den entsprechenden Verzeichnissen in /etc von mir durchgeführten Konfiguration. Das Ziel ist auch eine Bündelung aller für die Konfiguration notwendigen Dateien möglichst an einem Ort, von der ich mir bei der Installation eines ‚neuen‘ Servers nach einem Betriebssystem-Upgrade eine maßgebliche Vereinfachung verspreche. Ich empfehle natürlich hier, sich besser an die Standards zu halten. Aber wenn man das nicht tut und einen eigenen Weg einschlägt, sollte man sich dessen und der möglichen Konsequenzen durchaus bewusst sein.

Namensregeln für Postfachname bzw. Email-Adresse:

Auch wenn ich bei den Zertifikaten einen eigenen Weg gegangen bin, so bedeutet das nicht, dass man das gedankenlos so einfach an jeder Stelle machen darf oder tun sollte. Selbst wenn es sich hier um einen privaten Email-Server handelt und lokale Email-.Adressen über ein Per-Sender-Replacement zu einer FQDN-Adresse ersetzt werden, sollte dennoch der lokale Anteil der Emailadresse immer nur mit gültigen Zeichen eingerichtet werden. Einen guten Überblick über erlaubte und nicht-erlaubte Zeichen in Email-Adressen enthält die folgende Web-Site, die unbedingt einen tiefen Blick wert ist:

https://www.jochentopf.com/email/chars.html

Existierende Domains:

An mehreren Stellen in dieser Dokumentation nenne ich oder verwende ich in meinen Beispielen auch real existierende Domains, und zwar u.a. 1&1, Strato, GMX, Gmail und Web.de. Ich habe diese Namen willkürlich gewählt und mir ist bewusst, dass das nicht wirklich ‚*astrein*‘ ist. Aber gerade bei diesem wirklich komplexen Thema war es mir auch wichtig, bei bestimmten Sachverhalten deutlich zu machen, was ich meine, um verwechslungsfrei unterscheiden zu können, auf was ich mich bei bestimmten Sachverhalten überhaupt beziehe.

‚example.com‘ kann alles mögliche sein. Und wenn man ‚example.com‘ an verschiedenen Stellen verwendet, z.B. einmal als Hoster meiner Domain, einmal als Mail-ISP, wenn ich vielleicht gerade GMX meine, dann weiß nachher niemand mehr, worüber überhaupt gesprochen wird. Keiner kennt ‚example.com‘, man kann es eigentlich mit nichts assoziieren, ein Zusammenhang ergibt sich vielleicht aus meinem Text – aber auch nur, wenn es mir mit großem Geschick gelungen ist, den Zusammenhang offensichtlich herzustellen. Spreche ich hingegen von Gmail, GMX oder Web.de, weiß wirklich jeder sofort, was gemeint ist, ohne dass ich Anstrengungen für Erklärungen unternehmen muss. Ich habe mich deshalb entschieden, die Eindeutigkeit in meinen Erklärungen nicht durch Allgemeinplätze zu opfern, die alles und nichts bedeuten können und stattdessen in vielleicht nicht einwandfreier Art und Weise diese Domains in meine Erklärungen einzubeziehen.

Änderungsprotokoll

Datum	Seite	Änderung
24.10.2017	2	Inhaltsverzeichnis aktualisiert
	16	Rechte für Verzeichnis /media/SSD/Mail/Sieve korrigiert
	17	Kommentar zu den Rechten eingefügt
	68	Änderungsprotokoll angefügt
18.12.2017	24	Kapitel 9: Implementierung Post-Login-Script in dovecot.conf (siehe auch Kapitel 14, Getmail-Eventhandler als Ersatz für vorheriges TCP-Send)
	33 - *	Verzeichnis auf '*sprechenden*' Namen geändert Alt: /Postfix/sik → Neu: /Postfix/rules
	19 - 20 33 - *	Filenames Postfix-Regeln auf '*sprechende*' Namen geändert generic_replace_from → check_generic_replace_from sender_filter → check_sender_filter - → check_sender_relayaccess virtual_alias_maps → check_local_alias_maps transport_maps → check_local_transport_maps sender_relay → get_sender_relayhost sasl_passwd → get_relayhost_passwd header_checks → remove_header_sensitives
	36 - 40	Umfassende Behandlung der smtpd-restrictions eingefügt
	42	Kapitel 11: main.cf = kein eigenes Postfix-Zertifikat und Key mehr, Verweis auf: smtpd_tls_cert_file = /etc/dovecot/certs/dovecot.cert smtpd_tls_key_file = /etc/dovecot/certs/dovecot.key
	32 - 44	Kapitel 11: Postfix-Einrichtung redaktionell überarbeitet
	45 + 46	Kapitel 12: Getmail an neues Handling mit Dovecot-Post-Login-Script angepasst.
	53	Kapitel 14: Getmail-Eventhandler nach Implementierung von Dovecot-Post-Login-Script radikal vereinfacht.
	57 - 63	Kapitel 15: aufgetrennt in Kapitel 15 und 17 und an Änderungen durch neuen Getmail-Eventhandler angepasst Kapitel 15,16 und 17 neu geordnet
01.02.2019	 30	Schreibfehler (Ursache Copy/Paste) korrigiert doveadm pw : Single-Quotes und Verify hinzugefügt (siehe Man-Page *"You should enclose the password hash in single quotes"*)
07.09.2019	*	Korrekturen im Text
09.10.2019	25	dovecot.conf: Socket-Permissions für stats-reader/writer nach der Migration auf Debian Buster mit einer Berechtigung für die Gruppe 'vmail' gesetzt.

<table>
<tr><td></td><td>*</td><td>Überarbeitung der Berechtigungen von bisher (faktisch unnötigen) user-individuellen Rechten auf den zentralen Mailuser 'vmail'. Die individuellen Rechte hatten keine Vorteile, jedoch den Nachteil Wartungsarbeiten oder Änderungen unnötig zu verkomplizieren und sicherheitsrelevante Überprüfungen zu vernebeln.</td></tr>
<tr><td></td><td>-</td><td>/etc/environment ist offensichtlich nicht mehr nötig und wurde hier aus den Maßnahmen für Settings entfernt.</td></tr>
<tr><td></td><td>*</td><td>Mailname von 'raspi3.mail' auf 'mail' geändert, siehe Hinweise S. 45, Abs.2</td></tr>
<tr><td></td><td>*</td><td>Harmonisierung zu den in den Beispielen verwendeten IP-Adressen in den Artikeln security, openvpn, cameventctl zur Verbesserung der Zusammenhang-Darstellung.</td></tr>
<tr><td>05.01.2020</td><td>56-57</td><td>getmail-eventhandler:
- Erkennung Linux-User überarbeitet
- Plausibilitätsprüfungen überarbeitet
- Fehlermeldungen (journald) verbessert</td></tr>
<tr><td>04.03.2020</td><td>63</td><td>Fehler bei User-Kontextwechsel (su) behoben: Neu=vmail</td></tr>
<tr><td>09.06.2020</td><td>60</td><td>Erklärung zu Namenskonflikt in Grafik eingefügt.</td></tr>
<tr><td>20.08.2020</td><td>61</td><td>Fehlerbehandlung für „no suitable linux-user found in alias_maps“ eingefügt.</td></tr>
<tr><td>15.11.2021</td><td>12</td><td>SASL-Moduls zur Installation hinzugefügt</td></tr>
<tr><td></td><td></td><td>Migration nach Debian Bullseye (neues Setup) erfolgreich</td></tr>
<tr><td>25.04.2022</td><td>57</td><td>Fehlermeldung im Journal behoben:
<code>Apr 25 07:29:47 raspi3 getmail_eventhandler[]: process canceled! no suitable linux-user found in alias_maps: Virtuser=username@mail</code>

Ursache:
Unterschiedliche Mail-Clients (Linux, Android, Windows) praktizieren anscheinend unterschiedliche IMAP-Anmeldungen, manche mit „@mail“ hinter dem Usernamen, manche ohne.

Lösung:
getmail_eventhandler: entfernen von „@mail“ bei der Identifizierung des virtuellen Users als realer Linux-User in „alias_maps“

Codezeile:
<code>alt: VirtUser=$(echo "$USER" | sed 's/^[\t]*//; s/[\t]*$//')</code>
<code>neu: VirtUser=$(echo "$USER" | sed 's/@mail//; s/^[\t]*//; s/[\t]*$//')</code></td></tr>
<tr><td>25.07.2022</td><td>18-22</td><td>Neues Kapitel Openssl / TLS eingefügt. Ersetzen der einfachen Zertifikate durch Self-Signed Zertifikate</td></tr>
<tr><td></td><td>28-30</td><td>Aktualisierung der dovecot.conf, mit dem Ziel den Login-/Connect-Prozess von STARTTLS (Port 143) auf SSL/TLS (Port 993)umzustellen und mindestens TLSv1.2 als obligatorisch zu verlangen.</td></tr>
<tr><td></td><td>47-48</td><td>Aktualisierung der Postfix main.cf auf die neuen Zertifikate und erweiterte Sicherheitseinstellungen</td></tr>
<tr><td>01.08.2022</td><td>37, 43-44</td><td>Test- und Diagnoseverbindungen auf den Server für Dovecot und Postfix mit Openssl über Commandline als Ersatz für Telnet eingefügt. Mit der derzeitigen Beschränkung, Verbindungen nur noch über TLS und ohne Plaintext-Attribute zuzulassen, ist eine Verbindung über Telnet nicht mehr möglich.</td></tr>
<tr><td>01.09.2023</td><td></td><td>Migration nach Debian Bookworm (Testphase bis 11/23)</td></tr>
</table>

		Update auf Versionen:
		`dovecot-core      1:2.3.19.1`
		`dovecot-imapd     1:2.3.19.1`
		`dovecot-lmtpd     1:2.3.19.1`
		`dovecot-sieve     1:2.3.19.1`
		`postfix           3.7.6-0`

	49	Main.cf, weil es keine alten Geräte zu unterstützen gibt, Level von 2 auf 3.6 angehoben `compatibility_level = 3.6`